WYDAWNICTWO NOT SIGMA

P. 1877/90

Cena 5800 zł

9

1990

informatyka

Kompilatory firmy **Jensen & Partners International** już dostępne w Polsce!

TopSpeed Modula-2
TopSpeed C

Produkty firmy JPI są od dwóch lat światową rewelacją. Bezkonkurencyjna jakość generowanego kodu, w pełni zintegrowane środowisko ze znakomitym edytorem i wszelkimi narzędziami potrzebnymi profesjonalnemu programiście – to czynniki, które złożyły się na końcowy sukces tej firmy.

Wszystkie kompilatory – w przygotowaniu: TopSpeed Pascal, TopSpeed C + + (1990) i TopSpeed Ada (1991) – pracują w tym samym środowisku, co umożliwia realizację dużych projektów przy użyciu różnych języków i bez konieczności poznawania wielu, różniących się od siebie narzędzi. Dzięki temu można kompilować programy napisane na Turbo Pascalu firmy Borland.

Zintegrowane środowisko kompilatorów JPI to m.in.:

* edytor (9 okien, zaznaczanie bloków kolumnami, tablica ASCI, kalkulator, możliwość definiowania klawiszy, ...),
* pomoc w formie HyperTextu,
* SmartLinker wybierający z modułów tylko używane procedury,
* Visual Interaktiv Debugger pozwalający na uruchamianie i śledzenie programów (także współbieżnych) - interfejs do programów uruchomieniowych innych firm,

(programowanie obiektowe...)
(100% ANSI, programowanie współbieżne...)

* debugger „post-mortem"(!)
* biblioteka z pełnym kodem źródłowym,
* grafika własna lub możliwość korzystania z bibliotek firmy Borland,
* TopSpeed: Assembler, Profiler, Disassembler,
* PROM Locator (sterowniki przemysłowe),
* i wiele innych użytecznych narzędzi dla profesjonalnych programistów.

Wyżej wymienione oprogramowanie oferujemy w wersjach: profesjonalnej (DOS), profesjonalnej rozszerzonej (DOS) oraz profesjonalnej rozszerzonej (OS/2).

Jako dystrybutor oprogramowania firmy JPI w Polsce zapewniamy całodobowy serwis techniczny. Otrzymacie Państwo również licencję na sprzedaż programów napisanych przy użyciu zakupionych kompilatorów.

Proszę do nas dzwonić lub pisać w celu otrzymania dodatkowych informacji. Na życzenie wysyłamy pełną specyfikację przedstawionych pakietów. Oferujemy również dodatkowe biblioteki znanych firm światowych (bazy danych, grafika, sieci, generatory aplikacji itp.)

MERCURIA LTD
ul. Powstańców Śląskich 95
53-332 Wrocław

telefon (071) 60 56 79
telefaks (071) 61 67 11
teleks (071) 24 91 opg pl

0/11/90

informatyka

nr 9 1990
Wrzesień
Rok wydania XXV

Miesięcznik

ISSN 0542-9951. INDEKS 36124

KOLEGIUM REDAKCYJNE:

mgr Jarosław DEMINET,
dr inż. Wacław ISZKOWSKI,
mgr Teresa JABŁOŃSKA,
(sekretarz redakcji)
Władysław KLEPACZ
(redaktor naczelny)
dr inż. Wojciech MOKRZYCKI,
mgr inż. Jan RYŻKO,
mgr Hanna WŁODARSKA.

PRZEWODNICZĄCY
RADY PROGRAMOWEJ:

Prof. dr hab.
Juliusz Lech KULIKOWSKI

WYDAWCA: Wydawnictwo Czasopism i Książek Technicznych SIGMA NOT
Spółka z o.o.
ul. Biała 4,
00-950 WARSZAWA
skrytka pocztowa 1004

Redakcja: 01-552 Warszawa,
Pl. Inwalidów 10 p. 128, 136
tel. 39-14-34

Materiałów nie zamówionych redakcja nie zwraca

DRUK:
PRASOWE ZAKŁADY GRAFICZNE
RSW „PRASA-KSIĄŻKA-RUCH
ul. Dworcowa 13,
85-950 BYDGOSZCZ
zam. 2159/90
Obj. 4,0 ark. druk. Nakład 5250 egz.

Cena egzemplarza 5800 zł

W sprawach ogłoszeń prosimy zwracać się bezpośrednio do Redakcji

W numerze:

	Strona
Język Lisp i elementy sztucznej inteligencji (1) – *Zdzisław Wrzeszcz*	1
Historyczna szansa polskiej informatyki – *Andrzej Górecki*	5
Rozproszone przetwarzanie w grafice i transputery – *Krzysztof Zieliński*	7
Pomiar czasu obliczeń na IBM PC – *Zbigniew Cylkowski*	11
Automatyczna standaryzacja słów tekstu w języku polskim – *Piotr Żychowski*	15
Fizyczne struktury systemów baz danych (2). Techniki mieszania – *Miłosz Muszyński*	17
Przemysłowe systemy czasu rzeczywistego (2). Struktura i funkcje biblioteki DIALOGUE – *Mariusz Postół*	22
Z kraju	26
Ocena stanu polskiej informatyki	
/unix	28
The Design and Implementation of the 4.3BSD Operating System	
Systemy otwarte i ich wrogowie, czyli co dalej z UNIX-em (1). Rozłam w UNIX-ie	
zBITki	III okł.
1990 PRINT „BASIC ma 26 lat"	

W najbliższych numerach:

- Jarosław Deminet opisuje sposób działania komputerów i systemów komputerowych w Kongresie USA.
- Ewa Grabska, Grażyna Hliniak i Jacek Lembas charakteryzują zastosowanie grafów kompozycyjnych do konstrukcji strukturalnego edytora graficznego.
- Wojciech Mokrzycki rozpoczyna cykl artykułów na temat maszynowego postrzegania.
- Wiesław Lucjan Nowiński zajmuje się problemami współbieżnej rekonstrukcji obrazów.
- Andrzej Materka, Włodzimierz Górnisiewicz i Mariusz Kukuła opisują mikrokomputerowe systemy przetwarzania obrazów.
- Katarzyna Lange poświęca artykuł systemowi poczty elektronicznej – jednej z usług oferowanych przez oprogramowanie sieciowe Advanced Netware 286 firmy Novell dla sieci ArcNet.

Warunki prenumeraty

Przyjęcie prenumeraty – wyłącznie na podstawie dokonanej wpłaty na drukach dostarczanych dotychczasowym prenumeratorom przez Wydawnictwo, lub nowym – po uprzednim zgłoszeniu zapotrzebowania (pisemnie lub telefonicznie) w Zakładzie Kolportażu Wydawnictwa.

Blankiet wpłaty – powinien zawierać następujące informacje: dokładna nazwa i adres (z kodem pocztowym) zamawiającego, tytuły zamawianych czasopism, ich liczbę i okres prenumeraty.

Wpłata – zgodnie z podanymi cenami należy dokonać w banku lub w UPT na konto podane na naszym blankiecie, tj.:
Państwowy Bank Kredytowy III O/Warszawa nr: 370015-1573-139-11

Prenumeratorzy zbiorowi – osoby prawne obowiązują blankiety „Wpłata-Zamówienie". Cena normalna.

Prenumeratorzy indywidualni – osoby fizyczne obowiązują blankiety typu przekazy dla wpłat na rachunki bankowe. Cena normalna.

Prenumerata ulgowa – zgodnie z podaną ceną ulgową przysługuje wyłącznie osobom fizycznym, będącym członkami SNT, studentom i uczniom szkół zawodowych. Uczniowie szkół ogólnokształcących mogą zamówić w prenumeracie ulgowej tylko miesięcznik „Aura".
Uwaga: w podanym okresie prenumeraty można zamówić tylko po jednym egzemplarzu z każdego tytułu.

Prenumerata ze zleceniem wysyłki za granicę – cena prenumeraty jest dwukrotnie wyższa od ceny normalnej.
Należy podać dokładny adres odbiorcy za granicą.

Terminy przyjmowania prenumeraty:
do 10 listopada na I, II, III, IV kwartał następnego roku
do 28 lutego na II, III, IV kwartał br.
do 31 maja na III, IV kwartał br.
do 31 sierpnia na IV kwartał br.
Zmiany w prenumeracie, np. zmiana liczby tytułów, liczby egzemplarzy, rezygnacja z prenumeraty, można zgłaszać tylko w podanych terminach z mocą obowiązującą od następnego kwartału.

Egzemplarze archiwalne (z lat ubiegłych)
Można nabyć za gotówkę w Klubie Prasy Technicznej w Warszawie, ul. Mazowiecka 12 (tel. 26-80-16) lub zamówić pisemnie w Zakładzie Kolportażu, Dział Handlowy, 00-950 Warszawa skr. poczt. 1004 (tel. 40-37-31) na rachunek lub za zaliczeniem pocztowym.

Informacji o prenumeracie udziela: Zakład Kolportażu Wydawnictwa SIGMA-NOT Spółka z o.o., 00-716 Warszawa, ul. Bartycka 20, skr. 1004. Telefony: 40-00-21 wewn. 293, 295, 299 lub 40-30-86, 40-35-89.

Wstępna cena jednego egzemplarza na 1990 rok: normalna – 5800 zł.
ulgowa – 1160 zł.

Wartość prenumeraty (w zł):
Normalna: kwartalna – 17 400, półroczna 34 800, roczna 69 600
Ulgowa: kwartalna – 3 480, półroczna 6 960, roczna 13 920

Uwaga: W przypadku zmiany cen w okresie objętym prenumeratą, prenumeratorzy zobowiązani są do dopłaty różnicy cen.

P.1877/90

ZDZISŁAW WRZESZCZ
Instytut Maszyn Matematycznych
Warszawa

Język Lisp i elementy sztucznej inteligencji (1)

Lisp jest językiem podstawowym w dziedzinie badań nad sztuczną inteligencją (AI). W badaniach tych można wyróżnić dwa kierunki: w obu występuje umysł ludzki i komputer. W pierwszym przypadku umysł jest obiektem badanym, a komputer narzędziem badawczym, w drugim zaś komputer jest celem badań, a umysł ludzki jest modelem zachowania tej maszyny. Podejście pierwsze jest typowe dla psychologii poznawczej; poszukuje się tu wyjaśnienia procesów myślowych (zwłaszcza procesów twórczych) u człowieka za pomocą komputerowej symulacji tych procesów. W podejściu drugim, w którym celem badań jest komputer rozwiązujący problemy, model intelektu staje się wzorem do naśladowania.

Pierwszą propozycję języka Lisp sformułował John McCarthy w końcu lat pięćdziesiątych. Celem McCarthyego był język do przetwarzania struktur symbolicznych. Opracowany przez niego model takich obliczeń jest kompletnym opisem syntaktyki i semantyki języka. Nazwa Lisp pochodzi od angielskich słów *LISp Processing*, bo właśnie lista stanowi podstawę do formułowania zarówno danych jak i programów.

Od pierwszych badań nad sztuczną inteligencją Lisp stał się podstawowym językiem do formułowania problemów, dzięki wbudowanym mechanizmom manipulowania strukturami symbolicznymi. W miarę rozwoju badań nad sztuczną inteligencją powstało wiele formalizmów wysokiego poziomu opartych na listach. Formalizmy te wiążą się z określonymi metodami reprezentowania wiedzy w systemie AI i metodami sterowania procesem wnioskowania. Chodzi tu, na przykład, o regułę produkcji i system produkcji, ramę i system wzorców. Lisp jest idealnym językiem do implementowania tych mechanizmów.

Celem cyklu artykułów jest zapoznanie Czytelnika z mechanizmami sztucznej inteligencji, tj. z jej metodami i sposobem implementowania tych metod. Oprócz języka Lisp istnieją również i inne języki AI. Języki te (Prolog [3], grupa języków obiektowych [6]) są jednak dostosowane do określonych metod, a Lisp jest językiem uniwersalnym.

ZARYS ROZWOJU DZIEDZINY I JĘZYKA

Inteligencja jest tym, co kieruje działaniami człowieka. Znaczenie inteligencji dostrzeżono już w czasach antycznych. Z inteligencją związało się wiele mitów i legend, inteligencja absorbowała uwagę filozofów i pisarzy, powstawały rozprawy i dzieła literatury pięknej poruszające aspekty natury moralnej (inteligentny człowiek a Bóg itp.). Jednak aspekt **logiczny** inteligencji pojawia się dopiero u Arystotelesa. W swoim dziele „*Fizyka*", filozofię natury przedstawił jako studium rzeczy zmieniających się, np. kamień zmienia się w rzeźbę w wyniku pracy twórczej rzeźbiarza. Wprowadzone przez Arystotelesa rozróżnienie materii od formy stanowi filozoficzną podstawę dla obecnych koncepcji rozwoju nauki komputerowej; dotyczy to mianowicie koncepcji **przetwarzania symbolicznego** i **abstrakcyjnych struktur danych**.

Po Arystotelesie było wielu myślicieli rozwijających logiczny wątek inteligencji (Kartezjusz, Leibnitz, Pascal i inni), m.in. G. Boole (1815–1864). Ten wybitny filozof i matematyk angielski stworzył algebrę zmiennych dwuwartościowych (zmiennych logicznych), a następnie logikę nazwaną od jego nazwiska logiką booleowską, która stanowi podstawy matematyczne dzisiejszych komputerów. Boole też dostrzegł potrzebę opracowania „matematyki ludzkiego umysłu".

Sto lat później do myśli tej nawiązał Alan Turing (1912–1954), pracując nad pierwowzorem dzisiejszego komputera. Rozważał on realizację operacji zarówno na liczbach, jak i na wartościach logicznych. Jednak za początek prac nad sztuczną inteligencją przyjmuje się rok 1956 [5], datę konferencji w Dartmouth (USA), zorganizowanej przez Johna McCarthyego i Marvina Minskyego. Na konferencji tej McCarthy zgłosił swoją słynną propozycję odbycia przez dziesięciu naukowców dwumiesięcznego studium, którego celem było takie sformułowanie podstawowych zdolności umysłu ludzkiego (percepcja, pamiętanie, uczenie się, interpretacja, wnioskowanie itd.), by można było zbudować maszynę zdolną symulować te procesy. Wtedy też po raz pierwszy pojawiło się określenie *artificial intelligence*, w skrócie AI. W tekstach polskich używa się terminu **sztuczna inteligencja** choć może bardziej odpowiedni jest tu sztuczny intelekt.

O rozwoju danej dziedziny decyduje również rozwój technologii: bez odpowiednich narzędzi nie można zweryfikować koncepcji, a wtedy najlepsze pomysły są zwykłą literaturą. Połączenie więc myśli i technologii wyznacza etapy rozwoju sztucznej inteligencji (tabela).

Geneza i chronologia prac nad sztuczną inteligencją

Okres	Ważniejsze działania i wyniki
Początek badań: do 1939 r.	Prace nad logiką formalną Prace w dziedzinie psychologii poznawczej
1945–1954 Przygotowania teoretyczne	Prace konstrukcyjne nad komputerami Praca A. M. Turinga [5] Praca H. Simona [6] Praca N. Wienera [7]
1955–1960 Początek prac doświadczalnych	Komputery I generacji (IBM 704, 709 7094; w Polsce ZAM-2) Newell i Simon – opracowanie języka IPL-I Newell, Shaw i Simon: praca nad systemem GPS (*General Problem Solver*)
1961–1970 Prace nad systemami rozwiązywania problemów ogólnego przeznaczenia	Praca Newella i Simona [1] J. McCarthy: LIPS [9] Metodologia poszukiwania rozwiązań zadowalających Heurystyka i jej znaczenie w rozwiązywaniu problemu Robotyka Opracowanie systemu DENDRAL (Stanford University)
1971–1980 Systemy wyspecjalizowane, systemy ekspertowe praktyczne zastosowania AI	Komputerowy doradca lekarza – system MYCINE (Stanford); HEARSAY – system do analizy i generowania zdań języka naturalnego; EMYCIN – narzędziowy system ekspertowy; Prolog – język oprogramowania logicznego, R. Kowalski [12], A. Colmerauer [10];
1981 Rozpoczęcie prac nad komputerami AI [9]	Moto-Oka (Japonia) – program prac nad komputerami V generacji; Program ALVEY (W. Brytania) i ESPRIT (Francja)

Twórcą Lispu również jest John McCarthy. Lisp był początkowo językiem prostym. Zawierał jedynie kilka funkcji niezbędnych do operowania elementami listy, do definiowania nowych funkcji, predykat równości oraz funkcji do oceny wywołań funkcji. Środkiem do sterowania programem była rekurencja oraz pojedynczy warunek. Te prymitywne środki stanowiły podstawę do definiowania bardziej złożonych funkcji, które z biegiem czasu stawały się częścią języka Lisp. Tak powstały specjalne funkcje arytmetyczne – wejścia, wyjścia i sterowania programem – tworzące pełnowartościowy język programowania nie tylko dla struktur symbolicznych. Jednak w związku z tak ukształtowanym rozwojem powstały liczne dialekty języka, wśród nich

- ZETALISP dla komputerów lispowych firmy Symbolics,
- INTERLISP do pracy w trybie dialogowym (komputer Xerox),
- MACLISP do pracy w systemie operacyjnym UNIX
- Common LISP dla komputerów firmy DEC.

Common Lisp jest wynikiem prac inspirowanych przez DARPA [9] i ma pełnić funkcję standardowego dialektu Lispu. Popularyzacja Common Lispu przebiega pomyślnie, zwłaszcza po opracowaniu wielu translatorów w wersjach dla mikrokomputerów. Ta wersja Lispu będzie przedmiotem artykułu.

SYMBOLICZNA REPREZENTACJA PROBLEMU

Mówi się o problemie, gdy pragnąc czegoś nie wiemy jak to osiągnąć. Przedmiot pragnień może mieć formę materialną lub abstrakcyjną. Próbując rozwiązać problem, podejmuje się pewne akcje, z których część powoduje zmianę stanu przedmiotów ze świata problemu, inne natomiast dotyczą oceny stanu przedmiotów problemu i ich relacji, pamiętania stanów i relacji, podejmowania decyzji o kolejnych akcjach, porównywania wyników kolejnych akcji z celem itd.

Drugi rodzaj akcji jest realizowany przez umysł człowieka: sterują one akcjami pierwszego rodzaju. Zgodnie z hipotezą Newella i Simona [7] działanie umysłu można przedstawić jako działanie pewnego inteligentnego systemu przetwarzania informacji (ang. *intelligent processing system* – IPS).

System IPS zawiera następujące składniki:
- receptory bodźców zewnętrznych;
- zbiór symboli fizycznych; każdy element tego zbioru jest reprezentantem obiektu ze świata problemu; mogą to być obiekty fizyczne i relacje między nimi wynikające z określonych procesów fizycznych lub obiekty abstrakcyjne i związane z nimi zależności; symbole fizyczne obiektów i relacji tworzą pewne asocjacje reprezentujące problem;
- pamięć ulotną do przechowywania bieżącego stanu rozwiązania problemu;
- pamięć trwałą do gromadzenia asocjacji fizycznych reprezentujących metody rozwiązywania problemu;
- stały i ograniczony zbiór procesów przedstawionych do
  tworzenia nowych symboli fizycznych,
  porównania symboli,
  wprowadzania nowych asocjacji;
- układ interpretujący, określający sekwencje procesów podstawowych.

Koncepcja Newella i Simona stanowi podstawę do formułowania dzisiejszych systemów sztucznej inteligencji. Nie jest to prawdopodobnie jedyna możliwa koncepcja, odnosi się ona jednak do wszystkich systemów AI w obszarze robotyki [2] wielkich baz wiedzy [4, 14] lub systemów ekspertowych [11, 13].

Pierwotne sformułowanie problemu w języku potocznym musi ulec transformacji, by problem mógł być zaakceptowany przez system AI. Oba te sformułowania powinny więc mieć zbliżoną postać. Oto przykłady prostych zdań i ich reprezentacja.

1) *Jan posiada dom*	*(posiada jan dom)*
2) *Blok pierwszy jest domem znajdującym się przy ulicy Głównej*	*(jest-to blok-1 dom)* *(jest-to glowna ulica)* *(znajduje-się blok-1 glowna)*

W kolumnie pierwszej zebrano zdania w języku naturalnym. W kolumnie drugiej – zdania w języku AI. W pierwszym zdaniu struktura obu rodzajów wyrażeń jest podobna, inaczej jest jednak w zdaniu drugim. Jedno zdanie złożone języka naturalnego wymaga do wyrażenia tej samej treści w języku systemu trzech wyrażeń.

Podstawowym wymaganiem jest jednoznaczność reprezentacji; dotyczy to zarówno obiektów, jak i predykatów. Jeśli np. w problemie występuje więcej Janów, to trzeba ich wyróżnić dodatkowym znakiem, np. *jan*-1, *jan*-2. Struktura wyrażenia po transformacji ma charakter funkcjonalny, co oznacza, że kolejność występowania obiektów w wyrażeniu jest ważna, chociaż na etapie formułowania języka AI kolejność ta jest dowolna. Ze względów historycznych przyjmuje się, że predykat występuje na pierwszej pozycji. Rezygnuje się przy tym ze znaków interpunkcji – funkcję tę pełnią nawiasy okrągłe. Pozwala to jednorodnie zapisywać zarówno zdania proste, jak i złożone.

PRZEGLĄD WŁAŚCIWOŚCI JĘZYKA LISP

Zasadniczą cechą Lispu jest jego ukierunkowanie na manipulowanie strukturami symbolicznymi, które są środkiem do reprezentowania problemu i jego rozwiązania. Inną ważną cechą Lispu jest niezwykła wprost elastyczność, umożliwiająca dostosowanie do rozwiązywanego problemu. Mimo tworzenia nowych języków AI, Lisp jest nadal podstawowym narzędziem profesjonalnym w dziedzinie sztucznej inteligencji.

Wyrażenie symboliczne

Wyrażeniami syntaktycznymi Lispu są wyrażenia zwane też s-wyrażeniami. Oznacza to, że zarówno struktura danych jak i program przyjmują jednorodną formę. S-wyrażenie może być *atomem* lub *listą*. Atomy są podstawowymi elementami syntaktycznymi (tj. takimi, które nie dają się rozłożyć na prostsze). Atom może być liczbą lub symbolem: zapis symbolu może składać się z liter, cyfr i następujących znaków

* – + / @ $ " & < >

Przykłady atomów
2.718281
6
imie-wlasne
predykat
nil

Druga postać *s*-wyrażenia, tj. lista, jest sekwencją atomów lub list, przedzielonych spacjami, całość zaś jest zamknięta nawiasami okrągłymi. Przykłady list:

(jan adam ania)
(3 6 2 10)
(*jest-to blok*-1 *dom*)
(* 6 8)
(+ (* 6 8) 100)
( )
(*obiekt* (*typ czlowiek*) (*stan zywy*))

Ostatnia lista jest listą zagnieżdżoną, a przedostatnia jest listą pustą i nosi nazwę *nil*. *Nil* jest więc zarazem listą i atomem – jest to jedyny taki przypadek.

Ocena wyrażenia symbolicznego

Wprowadzone z terminala wyrażenie symboliczne jest poddane działaniu programu interpretującego (nazywanego w skrócie interpreterem lub po prostu Lispem), wg schematu

read – eval – print loop

Wyrażenie jest więc czytane, oceniane i wpisywane na monitorze, przy czym czynność ta jest powtarzana (*loop*). Lisp ogłasza się określonym symbolem, np.

> >

po którym wpisuje się wyrażenia; po przeczytaniu s-wyrażenia

> > (*f a b*)

Lisp potraktuje pierwszy element listy jako funkcję, a elementy następne jako jej argumenty. Jeśli argument jest listą, to najpierw jest oceniona ta lista, a dopiero potem do uzyskanych wartości argumentów jest stosowana funkcja.

Wiązanie symbolu z wartością

Ocena przez Lisp liczby daje zawsze w wyniku jej wartość: ocena symbolu natomiast daje w wyniku wartość związaną z tym symbolem. Wiązania takiego dokonuje programista stosując funkcję Seto, na przykład:

> > (*setq a* 2)
2

Jako wynik oceny tego wyrażenia Lisp podał 2, czyli wartość drugiego argumentu: ważniejszy jednak w tym przypadku jest **efekt uboczny** działania funkcji *setq*, którym jest przypisanie wartości drugiego argumentu argumentowi pierwszemu, czyli symbolowi *a*. Po wywołaniu teraz symbolu *a*

```
> > a
2
```

Lisp poda wartość mu przypisaną, czyli 2. Bez takiego przypisania Lisp zgłosiłby błąd.

Sterowanie oceną s-wyrażenia

Nie zawsze pierwszy element listy powinien być traktowany jak funkcja. Aby temu zapobiec, stosuje się funkcję *quote*, która powoduje, że występujące za nią wyrażenie nie jest poddane ocenie – wynikiem funkcji jest to wyrażenie, np.:

```
> > (quote (xy a b))
(XY A B)
> > (quote (* 2 3 ))
(* 2 3)

Ale
> > (* 2 3)
6
```

Ze względu na częste występowanie funkcji *quote*, stosuje się jej zapis skrócony w postaci apostrofu stawianego przed wyrażeniem, które funkcja *quote* ma obejmować, np.:

```
> > '(xy a b)
(XY A B)
```

Tworzenie listy

W Lispie występują dwie funkcje, umożliwiające konstruowanie listy, a tym samym dowolnego wyrażenia. Pierwszą z nich jest funkcja *cons*, która powoduje dodanie elementu do listy, np.:

```
> > (cons 'a' (b c)
(A B C)
> > (cons 'anna' (jan ada))
(ANNA JAN ADA)
```

Jednak tworzenie w ten sposób listy z wielu elementów może być kłopotliwe, np.

```
> > (cons 'a (cons 'b (cons 'c nil)))
(A B C)
```

Funkcja *list* powoduje utworzenie listy z obiektów podanych jako argumenty tej funkcji, np.:

```
> > (list 'a 'b 'c)
(A B C)
```

Wydobywanie elementów listy

Oprócz konstruowania listy, istnieją też funkcje umożliwiające jej dekompozycję. Funkcja *car* pozwala wydobywać pierwszy element z listy, np.:

```
> > (car '(anna jan ada))
ANNA
```

Warto zwrócić uwagę, że do argumentu zastosowano funkcję *quote*; bez tego Lisp zgłosi błąd, jeśli symbolowi *anna* nie przypisano wartości.

Jednoargumentowa funkcja *cdr* daje w wyniku listę zawierającą wszystkie elementy z wyjątkiem pierwszego, np.:

```
> > (cdr '(jablko gruszka sliwka)
(GRUSZKA SLIWKA)
```

Funkcje *car* i *cdr* mogą być składane.

Funkcje predykatowe (predykaty)

Podstawowy zestaw funkcji Lispu obejmuje pewną dość liczną grupę funkcji, których zadaniem jest sprawdzanie spełnienia określonego warunku. Jeśli warunek ten nie jest spełniony, to wynikiem funkcji jest *NIL*, czyli fałsz. Jeżeli natomiast warunek jest spełniony, to wynikiem funkcji predykatowej jest jeden z obiektów:
- stały symbol *T* reprezentujący prawdę,
- atom,
- lista.

Funkcje predykatowe są niezbędne do sterowania programem, a ponadto stanowią ważną część rachunku logicznego.

Predykat *atom*. Argumentem tego predykatu jest funkcja lub lista. Lisp przekazuje w wyniku wartość *T*, jeśli argument jest atomem, a *NIL* w przeciwnym razie, np.:

```
> > (atom 'a)
T
> > (atom ' (a b c))
NIL
```

Predykat *zerop*. Argumentem predykatu jest numerał (wyrażenie numeryczne); jeśli jego wartością jest zero, to wynikiem predykatu jest *T*, w przeciwnym razie *NIL*, np.

```
> > (zerop 0)
T
> > (zerop 44)
NIL
```

Predykat *graterp*. Argumentem predykatoru jest lista. Lisp przekazuje w wyniku *T*, jeśli liczby w liście są ustawione w porządku malejącym, a *NIL* w innych przypadkach, np.:

```
> > (graterp '(6 5 4 3))
T
> > (graterp '(1 2 4 5))
NIL
```

Predykat *listp* służy do określania czy jego argumentem jest lista:

```
> > (listp 'x)
NIL
> > (listp 10)
NIL
> > (listp '(a b c)
T
> > (listp nil)
T
```

W związku z ostatnim wyrażeniem należy przypomnieć o podwójnej naturze symbolu *NIL* – jest i listą, i atomem).

Predykat *null*. Jeśli argument predykatu ma wartość *NIL*, to w wyniku otrzymuje się *T*, w przeciwnym razie wynikiem jest *NIL*, np.:

```
> > (null nil)
T
> > (null ( ))
T
> > (null 'x)
NIL
> > (null '(a b c))
NIL
```

Predykat ten często stosuje się w sterowaniu programem działającym na elementach listy.

Predykat *equal*. Jest to dwuargumentowa funkcja sprawdzająca czy jej argumenty są równoważnymi wartościami lispowymi, np.:

```
> > (equal 'x 'x)
T
> > (equal 'x 'y)
NIL
```

```
> > (equal '(a b c) '(a b c))
T
> > (equal '(a b c) '(a '(b) 'c))
NIL
```

Predykat *member*. Jest to dwuargumentowa funkcja, sprawdzająca, czy pierwszy argument jest zawarty w drugim, będącym listą. Jeśli warunek jest spełniony, to funkcja przekazuje w wyniku tę część drugiego argumentu, która zaczyna się tym samym elementem, co argument pierwszy:

```
> > (member 'a '(a b c))
(A B C)
> > (member 'b '(a b c))
(B C)
> > (member 'b '(a (b) c))
NIL
```

* * *

Na sztuczną inteligencję można patrzeć jak na dziedzinę badań mającą własne środki, tj. metody i narzędzia, przeznaczone do formułowania (a następnie realizowania) perspektywicznych systemów komputerowych. Zadaniem niniejszego artykułu pierwszego z cyklu, jest zapoznanie Czytelnika z występującymi tu pojęciami oraz z podstawami języka Lisp – zasadniczego środka do formułowania a następnie do realizacji systemu komputerowego.

LITERATURA

[1] Bishop P.: Fifth Generation Computers. J. Wiley, 1986

[2] Charniak E., McDermott D.: Introduction to Artificial Intelligence Addison-Wesley, Reading (MA), 1987

[3] Colmerauer A. et al.: Un systeme de communication home-machine en francais. Universite d'Aix-Marseille, 1973

[4] Gendarin G.: Relational Databases and Knowledge Bases. Addison-Wesley. Reading (MA), 1989

[5] McLennan J.: Principles of Programming Languages. Holt, Reinehart and Winston, 1987

[6] Meyer B.: Object-oriented Software Construction. Prentice-Hall, Englewood Cliffs (NJ), 1988

[7] Newell A., Simon H.: Human Problem Solving. Prentice-Hall. Englewood Cliffs (NJ), 1972

[8] Simon H. A.: The Science of the Artificial. MIT Press, 1981

[9] Steele G. L.: Common LISP The Language. Digital Press. 1986

[10] Turing A. A.: Computing Machinery and Intelligence. Mind., No. 59, pp. 433 460, 1950

[11] Waterman D.: A Guide to Expert Systems. Addison-Wesley. Reading (MA), 1986

[12] Wiener N.: Cybernetics. Scientific American, p. 179, 1948

[13] Wrzeszcz Z.: Podstawy implementacji systemów eksperckich na mikrokomputerze. Prace Naukowo-Badawcze IMM. Vol. 29, No. 1, 1987

[14] Wrzeszcz Z.: System z bazą wiedzy do gromadzenia informacji faktograficznej. I Krajowa Konferencja Naukowa. „Inżynieria wiedzy i systemy ekspertowe", Wrocław, 26 28 czerwiec 1990.

Docierają do Redakcji sygnały z całego kraju o trudnościach z dostaniem INFORMATYKI w kioskach Ruchu.

Pragniemy wyjaśnić Czytelnikom przyczynę tych kłopotów. Redakcja podjęła decyzję zmniejszenia liczby egzemplarzy skierowanych do sprzedaży kioskowej, gdyż Ruch pobiera bardzo wysoką prowizję za swą usługę, nie gwarantując prawidłowego rozdziału egzemplarzy w sieci kioskowej.

W związku z tym prosimy wszystkich zainteresowanych nabyciem INFORMATYKI o kontakt z redakcją (tel. 39-14-34) lub Działem Handlowym SIGMY (tel. 40-30-86), – będziemy państwu wysyłać egzemplarze za zaliczeniem pocztowym. Czytelnicy z terenu Warszawy mogą kupić poszczególne numery w Klubie SIGMY przy ul. Mazowieckiej 12, w księgarnii PP „Domu Książki" ul. Mokotowskiej 51/53 oraz w lokalu redakcji.

Jednocześnie przypominamy, że najpewniejszą formą otrzymywania INFORMATYKI jest prenumerata. Szczegóły jej zamawiania na II strone okładki.

INTERSOFT Sp. z o.o.

Nasze Biuro Handlu Zagranicznego poleca swoje usługi w zakresie eksportu-importu każdego towaru. Prowadzi sprzedaż profesjonalnego sprzętu komputerowego i peryferii.

1. JĘZYKI PROGRAMOWANIA, BIBLIOTEKI, GRAFIKA

Basic – programowanie w języku Basic cz. III	52 000 zł
Turbo Basic	183 000 zł
Proffesional Fortran + dodatki	144 000 zł
Fortran 77	132 600 zł
Fortran 80	65 300 zł
Język C – zastosowanie dla zaawansowanych	214 200 zł
Kompilator języka C	106 000 zł
System Turbo C V.1.5	79 000 zł
Turbo C biblioteka	150 000 zł
Turbo C podręcznik użytkownika	109 000 zł
Turbo Pascal	123 500 zł
Turbo Pascal V.4.0	277 000 zł
Turbo Pascal V.5.0	161 000 zł
Turbo Pascal V.5.5 dodatek do T.P.V.5.0	56 000 zł
Turbo Power Tools Plus	124 000 zł
Turbo Database Toolbox	33 000 zł
Modula 2 86	125 500 zł
Turbo Pascal 5.0 opis języka	189 000 zł
Turbo Pascal 4.0 (aut. Jan Bielecki)	153 000 zł

2. BAZY DANYCH, PAKIETY ZINTEGROWANE

Clipper do dBase III	30 000 zł
Clipper do dBase III +	131 000 zł
Clipper 86	158 000 zł
Clipper 87	212 700 zł
Clipper 87 do dBase III +	214 000 zł
dBase II i III – podręcznik dla zaawansowanych	205 000 zł
dBase III + instalacja pakietu relacyjnej bazy danych	22 000 zł
dBase III + zastosowanie	205 000 zł
dBase III + opis pakietu sieciowego	55 000 zł
dBase III + programowanie	126 000 zł
dBase III + poznawanie	111 000 zł
dBase III – poradnik encyklopedyczny	237 000 zł
dBase IV	249 000 zł
Framework II P	162 000 zł
Informix	261 000 zł
Lotus 1-2-3 + dodatek (komplet)	113 000 zł
Multiplan	66 000 zł
Pakiet C-ISAM	34 700 zł
CX – Fortran kompilator języka do systemu DOS2	114 300 zł
Plib 86/Plink 86 opis języka	47 000 zł
Statgraphics	91 000 zł
Programowanie systemów baz danych języka Clipper	107 000 zł
Kompilator CBasic tom. II	92 000 zł

3. SYSTEMY OPERACYJNE, PROGRAMY UŻYTKOWE

System operacyjny DOS cz. 1 i 2	155 000 zł
System operacyjny DOS 3.10	47 000 zł
System operacyjny DOS 3.20	181 000 zł
System operacyjny DOS 3.30	180 000 zł
System operacyjny DOS 4.0	202 000 zł
Eureka	64 000 zł
Norton Commander	63 000 zł
Polywindows	50 000 zł
Sidekick	44 000 zł

4. EDYTORY

Chl-2 Writer	31 000 zł
Personalny Edytor PE	68 300 zł
Wordstar 2000	142 000 zł

5. CAD-SIECI

AutoCad 2.17	137 000 zł
AutoCad	30 600 zł
Lanlink 4.0 sieć lokalna	80 000 zł
NOWELL, instalacja sieci (Arcnet)	9 000 zł
NOWELL, instalacja sieci	65 000 zł
NOWELL, podręcznik użytkowania sieci	116 000 zł
Or-Cad	134 000 zł

6. ROZMAITOŚCI

Przewodnik programisty	205 000 zł
Printer STAR NL-10 instrukcja	20 000 zł
Wprowadzenie do użytkowania komputerów osobistych klasy IBM PC	43 000 zł
Drukarka NX-16 instrukcja	106 000 zł
Drukarz	14 000 zł
Karta – system rozliczania kart drogowych	10 000 zł
Appsgen	18 000 zł

Zamówienia prosimy kierować pod adresem:
INTERSOFT Sp. z o.o.
00–496 Warszawa, ul. Krucza 16/22,
telefony: 28-44-81 w. 284, 352; 28-29-53,
teleks: 812414

0/7/90

ANDRZEJ GÓRECKI
Melbourne
Australia

Historyczna szansa polskiej informatyki

Dla nikogo chyba nie ulega wątpliwości, że obecny okres w Polsce jest okresem przełomowym. Stopniowe wprowadzanie wymienialności złotówki coraz silniej wywołuje w polskiej gospodarce prawidłowe działania sił rynkowych.

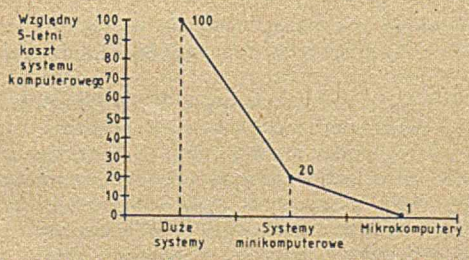
Zmiany te, jakkolwiek szybkie, nie przeniknęły jeszcze do wszystkich zakątków gospodarki i dlatego w wielu dziedzinach jest trudno przeprowadzić prawidłowy rachunek ekonomiczny. Moim zdaniem, to „przesączanie się" gospodarki opartej na prawidłowej wartości pieniądza potrwa około 2 lat. Należy jednak stwierdzić, że zawsze pozostaną pewne enklawy dysproporcji, co można zaobserwować w każdej, nawet najlepiej działającej gospodarce.

Brak w Polsce prawidłowych relacji wartości utrudnia (i jeszcze przez pewien czas będzie utrudniać) podejmowanie prawidłowych decyzji w wielu dziedzinach gospodarki, w tym również informatyki. Zważywszy, że nawet w gospodarkach zachodnich informatyka z trudem poddaje się analizie ekonomicznej, jest oczywiste, że podejmowanie decyzji na temat informatyki jest w Polsce szczególnie trudne.

Artykuł ten ma na celu zasugerowanie Czytelnikom, które trendy w dziedzinie informatyki należy obserwować, aby w okresie przejściowym zapewnić podejmowanie prawidłowych decyzji (z minimalnym elementem ryzyka). Okres taki będzie trwał do momentu, gdy gospodarka narodowa zostanie prawidłowo zbilansowana, stwarzając warunki pełnego zaufania do decyzji opartych wyłącznie na rachunku ekonomicznym.

W obecnej chwili informatyka polska znajduje się w specyficznym stanie. Duże systemy (JS) są przestarzałe, ograniczenia COCOM[1] utrudniają dostęp do nowoczesnego sprzętu, natomiast bardzo szybko rozpowszechniają się mikrokomputery (PC). Następuje całkowicie spontaniczna rewolucja w podejściu do przetwarzania danych. Rozważając sytuację informatyki musimy spojrzeć na trzy grupy sprzętu komputerowego:

- systemy duże (ang. *mainframes*),
- systemy średniej wielkości albo minikomputerowe (ang. *minicomputers*),
- systemy mikrokomputerowe (ang. *microcomputers*).



Rys. 1. Porównanie pięcioletniego kosztu posiadania systemu komputerowego w podstawowych grupach sprzętu

Według mojej oceny, obecna sytuacja w powyższych kategoriach sprzętu przedstawia się w Polsce tak, jak podano w poniższej tabeli. Natomiast porównanie powszechnie używanego w analizach ekonomicznych tzw. pięcioletniego kosztu posiadania systemu (ang. *cost of 5 year ownership*) w omawianych trzech kategoriach sprzętu komputerowego przedstawia rys. 1.

[1] Grupa robocza przodujących państw zachodnich zajmująca się ograniczeniami w przekazywaniu nowoczesnej technologii do krajów strefy wpływów radzieckich.

Obecna sytuacja sprzętowa w Polsce

Rodzaj systemu	Systemy dominujące w Polsce	Systemy nowoczesne	Przyczyny zacofania	Opóźnienie Polski w stosunku do świata
Systemy duże	IBM 360/370 JS ODRA	IBM 3090 Amdahl 59xx HDS EXxx	1) Koszt 2) COCOM	1965
Systemy minikomputerowe	PDP 11 SM	VAX Tandem	1) COCOM 2) Koszt	1970–1975
Mikrokomputery	IBM XT/AT	i386, i486 AT/MC/EISA Macintosh		1988

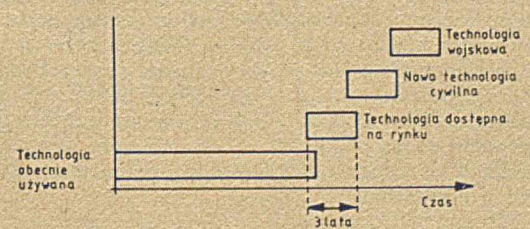
Jest sprawą niezmiernej wagi, aby polskie przedsiębiorstwa prawidłowo wykorzystały obecną sytuację. Uważam, że Polska jest w stanie zbudować informatykę (dla potrzeb przetwarzania danych) na poziomie przewyższającym to, co osiągnięto w wielu przodujących krajach zachodnich. Być może pogląd ten brzmi bardzo kontrowersyjnie, ale postaram się mój punkt widzenia uzasadnić.

Mówiąc o wysokim poziomie informatyki mam oczywiście na myśli osiągane rezultaty, a nie liczbę zainstalowanych systemów oraz ich wielkość i koszt. Jest bowiem sprawdzonym faktem, że efektywne i użyteczne systemy komputerowe nie muszą być kosztowne.

Poziom rozwoju informatyki na Zachodzie należy oceniać w czterech kategoriach:

- technologie wojskowe,
- nowe technologie cywilne,
- technologie dostępne na rynku,
- technologie aktualnie stosowane.

Relacje czasowe między tymi technologiami ilustruje rys. 2. Widoczne jest, że technologia wojskowa wyprzedza opublikowane nowe technologie cywilne o minimum 2 do 3 lat. Odstęp czasowy pomiędzy pojawieniem się nowej technologii w publikacjach, a początkami jej praktycznego zastosowania w gospodarce wynosi około 5 lat. Technologia używana w większości przedsiębiorstw zachodnich jest więc przeciętnie o około 10 lat w tyle w stosunku do tego, co czytamy w czasopismach na temat nowych rozwiązań komputerowych.



Rys. 2. Relacje czasowe rozwoju technologicznego informatyki

Ponieważ większość obecnie eksploatowanych komputerów została nabyta we wczesnych latach osiemdziesiątych, koszt ich zakupu znacznie przewyższa ceny płacone w dniu dzisiejszym za sprzęt o podobnej mocy obliczeniowej. Te kosztowne systemy są często przestarzałe, ale jeszcze nie mogą być zmodernizowane (wymienione) na bardziej nowoczesne ze względu na:

- zainwestowanie znacznych sum w oprogramowanie,

- znaczną pracochłonność, a więc wysoki koszt przeniesienia istniejącego oprogramowania na nowy system,
- fakt, że kosztowny sprzęt nie został jeszcze całkowicie spłacony, a więc nadal pozostaje własnością banku, który sfinansował transakcję jego kredytowego zakupu lub dzierżawy (ang. *leasing*).

Jeżeli nawet zachodnie przedsiębiorstwo zdecyduje się na zakup nowego komputera, to nadal używa posiadanego przestarzałego oprogramowania, co czyni wyżej wspomnianą dziesięcioletnią lukę zapóźnienia technologicznego jeszcze bardziej dotkliwą.

Szansa polskiej informatyki wynika z faktu, że gospodarka polska nie jest jeszcze tak nasycona kosztownymi systemami komputerowymi, jak to występuje w krajach zachodnich. Jedyny problem to uzyskanie funduszy na nowy system i, podobnie jak w krajach zachodnich, kwestia przeniesienia programów i danych na nowy komputer. Biorąc pod uwagę fakt, że osiągnięcie równoważnych rezultatów przy zastosowaniu dużych systemów komputerowych kosztuje 100 razy więcej niż przy użyciu mikrokomputerów, jest oczywiste, że przedsiębiorstwa w Polsce powinny skoncentrować się na rozwiązaniach opartych na mikrokomputerach.

W tych nielicznych przypadkach, gdy mikrokomputer jest niewystarczający (potrzeba np. pamięci dyskowej przekraczającej 500 MB, znacznej liczby pracujących współbieżnie terminali), należy używać minikomputerów. Minikomputery choć są około dwudziestokrotnie droższe w eksploatacji od mikrokomputerów, kosztują nadal około pięciu razy mniej, niż dostarczające identyczne rezultaty duże systemy komputerowe (patrz rys. 1.).

Obecnie różnice w mocy obliczeniowej między minikomputerami i mikrokomputerami stają się coraz trudniejsze do zauważenia. Nowoczesne mikrokomputery (oparte na mikroprocesorach klasy 386 lub 486), wyposażone w systemy operacyjne uwzględniające coraz bardziej popularny standard **UNIX**, mają już wszelkie cechy systemów minikomputerowych, włącznie z wielodostępem.

Możliwość równoczesnego dostępu do systemu mikrokomputerowego osiąga się także przez zastosowanie sieci lokalnych (LAN – *Local Area Network*), takich jak np. Novell, 3Com czy VINES. Sieci łączące 1000 lub więcej mikrokomputerów nie są już rzadkością w krajach zachodnich. Ponadto sieci takie można łatwo przyłączyć do minikomputerów i dużych systemów komputerowych, zapewniając dla wszystkich użytkowników łatwy, bezpośredni dostęp do wielkich zbiorów danych. Już dziś można nabyć system mikrokomputerowy za około 35 000 dol., który praktycznie niczym się nie różni od małego systemu minikomputerowego kosztującego około 200 000 dol.

W chwili obecnej na świecie istnieje wiele firm produkujących systemy minikomputerowe. Bardziej znane to: Digital Equipment Corporation (DEC) – wytwarzająca systemy VAX oraz w latach siedemdziesiątych systemy PDP; Data General – znana w Polsce z systemu NOVA; PRIME; Wang (system 2200); Tandem (system o podwyższonej niezawodności); IBM – systemy serii 36, 38 oraz AS/400 czy Hewlett-Packard (HP3000). Rynek minikomputerowy jest bardzo dynamiczny i często pojawiają się nowe produkty, o bardzo konkurencyjnych cenach i mało jeszcze znanych nazwach jak np. SEQUENT, PIRAMID, ADDS, SEQUOIA itp. Moim zdaniem tylko dwie firmy produkują minikomputery dające ich użytkownikowi gwarancję osiągnięcia sukcesu, tzn. możliwość rozbudowy systemu w miarę wzrostu potrzeb, łatwość programowania i eksploatacji oraz łączenia z innymi komputerami, stabilność rynkowa producenta (dostępność serwisu) itp. Są to firmy Tandem oraz DEC. Systemy minikomputerowe tych firm mogą być rozbudowane do tego stopnia, że osiągają moc obliczeniową przewyższającą to, co dotąd było osiągalne tylko za pomocą największych systemów firmy IBM.

Nie są mi znane ostatnie decyzje COCOM i z tego względu zakładam, że systemy firmy Tandem będą w Polsce nadal niedostępne. Jeśli chodzi o maszyny typu VAX, to widziałem ostatnio ogłoszenie tego sprzętu w prasie polskiej, ale czy jest on łatwo dostępny w Polsce, tego niestety nie wiem.

W warunkach polskich wybór jest więc jednoznaczny: należy uznać systemy VAX za strategiczną platformę sprzętową dla polskiej informatyki. Jeżeli systemy te będą trudno dostępne, należy używać systemów PDP lub ich radzieckich odpowiedników (SM), gdyż oprogramowanie z systemów PDP może być łatwo przeniesione na system VAX (jakkolwiek jest ono wg współczesnych wymagań bardzo nieefektywne). Systemy firmy DEC (VAX lub PDP) powinny być łączone w przedsiębiorstwie siecią lokalną.

Jeżeli polskie przedsiębiorstwa potrafią uniknąć pułapki dużych systemów, które wymagają ogromnych nakładów kapitałowych i są kosztowne w eksploatacji, to istnieje unikalna szansa przeskoczenia całego etapu rozwoju informatyki. Choć duże systemy komputerowe w krajach zachodnich są nadal szeroko stosowane, coraz silniejszy jest trend na rzecz ich wymiany na minikomputery (ang. *system downsizing*).

Pamiętając, że koszt eksploatacji systemu, to także koszt robocizny programistów i analityków systemowych (a tym samym czas potrzebny na rozwój nowych systemów i modernizację istniejącego oprogramowania), polska informatyka ma niepowtarzalną szansę nie tylko przeskoczenia całego etapu rozwoju. Możliwa jest także bardzo szybka realizacja tego skoku.

•

Od redakcji. Zamieszczony w nr 1/1990 nasz apel o nadsyłanie publikacji znalazł oddźwięk w dalekiej Australii, skąd w czerwcu otrzymaliśmy list oraz artykuł od naszego Czytelnika p. Andrzeja Góreckiego. Pan Górecki, absolwent Politechniki Warszawskiej, jest obecnie naczelnym dyrektorem sektora technologii informatycznej firmy James McEvans Ltd. w Melbourne. Jego australijska kariera zawodowa obejmuje realizację kilkudziesięciu projektów systemów informatycznych w różnych dziedzinach gospodarki tego wielkiego kraju.
Artykuł prezentuje interesujące, choć, jak przyznaje sam Autor, kontrowersyjne spojrzenie z poza kraju na problem likwidacji naszego zapóźnienia w dziedzinie komputeryzacji. Sądzimy, że artykuł ten, mimo pewnych uproszczeń w ocenie skali naszych potrzeb i możliwości, będzie stanowił tak potrzebną przeciwwagę dla dominującego pesymizmu na temat naszej aktualnej sytuacji.

Uwaga Twórcy, Konsultanci oraz zainteresowani programami komputerowymi!!!

Agencja Obrotu Oprogramowaniem

działająca przy Wydawnictwach Naukowo-Technicznych
to nowość na polskim rynku informatycznym

Główne formy naszej działalności to:
- wydawanie katalogu zawierającego informacje o aktualnie dostępnych na rynku programach oraz ogłoszenia na zamówienie klienta;
- rejestrowanie programów dla celów ochrony praw autorskich;
- pośredniczenie w sprzedaży oprogramowania;
- organizowanie przetargów na rozwiązania informatyczne.

Jeśli chcesz wybrać programy najbardziej odpowiednie dla Twojego przedsiębiorstwa, nie musisz już dzwonić do wielu firm ani jeździć po całym kraju w celu zapoznawania się z programami. My mamy je wszystkie w jednym miejscu a nasz adres jest do Twojej dyspozycji. Pomożemy Ci dokonać wyboru w sposób o jakim marzyłeś od dawna; nie ruszając się z biura naszej Agencji. Jeśli masz programy – efekt wielu miesięcy Twojej pracy, zarejestruj je u nas zanim piraci je skopiują, a my pomożemy Ci w ochronie Twoich praw autorskich.

Agencja Obrotu Oprogramowaniem
przy Wydawnictwach Naukowo-Technicznych
ul. Mazowiecka 2/4 pok. 206
00–048 Warszawa
tel. 26-71-71 wew. 304, telefaks 26-82-93

0/3/90

KRZYSZTOF ZIELIŃSKI
Instytut Informatyki
Akademia Górniczo-Hutnicza
Kraków

Rozproszone przetwarzanie w grafice i transputery

Zagadnienie rozproszonego przetwarzania informacji w grafice komputerowej staje się aktualne m.in. z powodu:
- **rosnących wymagań dotyczących rozdzielczości obrazu oraz jakości koloru,**
- **rozpowszechnienia zastosowań wymagających reprezentacji ruchu obiektów w naturalnej skali czasu, przy jednoczesnym zachowaniu odpowiedniej jakości obrazu,**
- **dążenia do wizualizacji wyników złożonych zadań obliczeniowych.**

Sprawia to, że powstające zadania przetwarzania przekraczają możliwości jednego procesora oraz przepustowości komunikacji z pojedynczymi blokami pamięci. Naturalne staje się w tej sytuacji poszukiwanie możliwości równoległego przetwarzania zadań graficznych, przez zastosowanie wielu niezależnie pracujących, lecz komunikujących się modułów obliczeniowych.

Celem artykułu jest rozważenie możliwości rozproszenia przetwarzania zadań graficznych oraz przedstawienie ich wykorzystania w systemach graficznych budowanych przy użyciu transputerów.

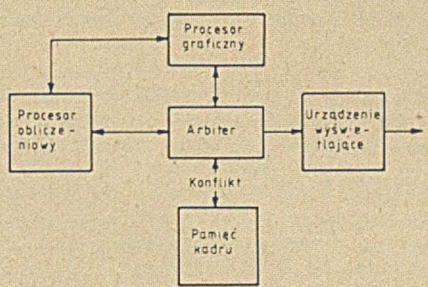
Rozważania dotyczące metod rozproszenia przetwarzania w grafice komputerowej mają charakter ogólny. Ograniczenie analizy możliwości implementacji tych metod do systemów transputerowych wynika z faktu, iż systemy te wydają się być obecnie najlepiej rozwinięte i najbardziej popularne w wielu ośrodkach naukowo-badawczych.

Analiza zadania przetwarzania informacji w grafice

Czynniki powodujące, że uzyskanie wysokiej klasy obrazu graficznego nie jest zadaniem prostym, są następujące:

1. Powszechnie stosowana zasada tworzenia obrazu na ekranie monitorów wymaga wielokrotnego (np. z częstotliwością 50 Hz) odtwarzania obrazu na ekranie. Stwarza to z kolei konieczność, aby z analogiczną częstotliwością była odczytywana pamięć kadru obrazu (skanowana) zawierająca informacje o poszczególnych pikslach ekranu. Zawartość pamięci kadru oraz jej struktura zależy od rozdzielczości ekranu oraz rozdzielczości koloru na ekranie. Za małą rozdzielczość ekranu uważa się najczęściej ekrany o wymiarach 640×480 piksli (VGA), natomiast za dużą – gdy liczba ta wynosi 1280×1024 piksli i więcej. Każdy z piksli jest reprezentowany w pamięci kadru przez jeden bajt, gdy paleta barw jest ograniczona do 256 kolorów, lub przez 3 bajty, z których każdy opisuje jedną z podstawowych barw niezależnie (czerwona, zielona i niebieska), co daje 16 777 216 kolorów. Należy zauważyć, że ograniczenie do 256 kolorów nie pozwala uzyskać kolorów zbliżonych do naturalnych. Tak więc pamięć kadru w przypadku obrazów dużej rozdzielczości, w kolorach prawie naturalnych, wynosi około 4 MB. Sam odczyt takiej pamięci wielokrotnie w ciągu sekundy wiąże się z bardzo wysokimi wymaganiami dotyczącymi czasu dostępu. Zazwyczaj, aby maksymalnie usprawnić operacje w pamięci kadru, stosuje się w tym celu specjalne procesory graficzne [3].

2. Pamięć kadru obrazu musi być aktualizowana, gdy chcemy uzyskać obraz ruchomy. Płynny ruch uzyskuje się przy około 20...30 aktualizacji zawartości pamięci kadru w ciągu sekundy. Powoduje to powstanie konfliktu w dostępie do pamięci kadru między procesorem przygotowującym nowe kadry, a procesorem graficznym (rys. 1). Powstający konflikt można zilustrować bardzo prostym przykładem. Niech będzie dany obraz złożony z 512×512 piksli, z których każdy jest opisany przez jeden bajt. Jeżeli pamięć kadru jest odczytywana słowami 4-bajtowymi, z czasem dostępu 200 ns na słowo, a częstotliwość skanowania wynosi 50 Hz, to łatwo obliczyć, że odczyt odbywa się co $20 \cdot 10^{-3}$ s, a czas odczytu pamięci kadru wynosi $13{,}1 \cdot 10^{-3}$ s w każdym cyklu. Wynika stąd, że na akcje procesora pozostaje $6{,}9 \cdot 10^{-3}$ s w każdym cyklu, co oznacza, że procesorowi pozostaje zaledwie 34% czasu na modyfikacje w pamięci kadru.



Rys. 1. Schemat logiczny współpracy procesora graficznego i procesora obliczeniowego

3. Obraz graficzny stanowi zazwyczaj rzut modelu zbudowanego w przestrzeni trójwymiarowej z dużej liczby wieloboków. Transformacje geometryczne, jak np. obrót, przesunięcie oraz skalowanie, takiego modelu są prowadzone typowo w czterowymiarowej przestrzeni współrzędnych jednorodnych i wymagają dużej liczby operacji zmiennoprzecinkowych (tabela). Do najczęściej wykonywanych operacji należy mnożenie macierzy czterowymiarowych. Ocenę liczby operacji potrzebnych do przetworzenia przykładowej sceny złożonej z 12 000 punktów, a wyrażonych w tysiącach operacji zmiennoprzecinkowych (KFLOP), podano w poniższej tabeli [4].

Szacunkowa liczba operacji zmiennoprzecinkowych potrzebna dla przetworzenia przykładowej sceny złożonej z 12 000 punktów

Przekształcenie obrazu	Liczba operacji zmiennoprzecinkowych (KFLOP)
Obrót, przesunięcie, skalowanie	300
Wycinanie pożądanego fragmentu obrazu i usuwanie zasłoniętych powierzchni (oknowanie)	72
Transformacja do współrzędnych ekranu	130
Cieniowanie	360
Interpolacja (zaokrąglanie płaskich powierzchni)	300
Łącznie	**1200**

Uwzględniając dane z tabeli i zakładając, że w ciągu sekundy ma być generowanych 25 kadrów (płynny ruch), otrzymuje się zapotrzebowanie na moc obliczeniową rzędu 30 MFLOPS (MFLOP/s). Jest to szybkość jeszcze nie osiągalna przez pojedyncze procesory i leżąca na granicy fizycznych ich możliwości (szacowanej na około 40 MFLOPS). Nawet samo przesłanie danych między procesorem a pamięcią leży poza możliwościami wielu komputerów.

Przedstawione rozważania pozwalają sprecyzować następujące **teoretyczne wymagania dotyczące systemu komputerowego do przetwarzania zadań graficznych:**

1) system powinien zapewniać bardzo dużą moc w sensie szybkości obliczeń z możliwością elastycznego zwiększania tej mocy,
2) powinna istnieć możliwość dowolnego zwiększania liczby wykonywanych operacji, w ramach aktualizacji pamięci kadru przez system obliczeniowy bez powodowania wzajemnego konfliktu między procesorami obliczeniowymi systemu;
3) operacja odczytu obrazu powinna mieć oddzielny dostęp do pamięci kadru, aby uniknąć konfliktu między procesorami obliczeniowymi a procesorem graficznym obsługującym monitor;
4) powinna istnieć możliwość dowolnego zwiększania rozdzielczości obrazu w sensie liczby pikseli oraz rozdzielczości koloru (liczby bitów na kolor), oczywiście możliwości wykorzystania tej cechy systemu są ograniczone dostępnością odpowiednich monitorów.

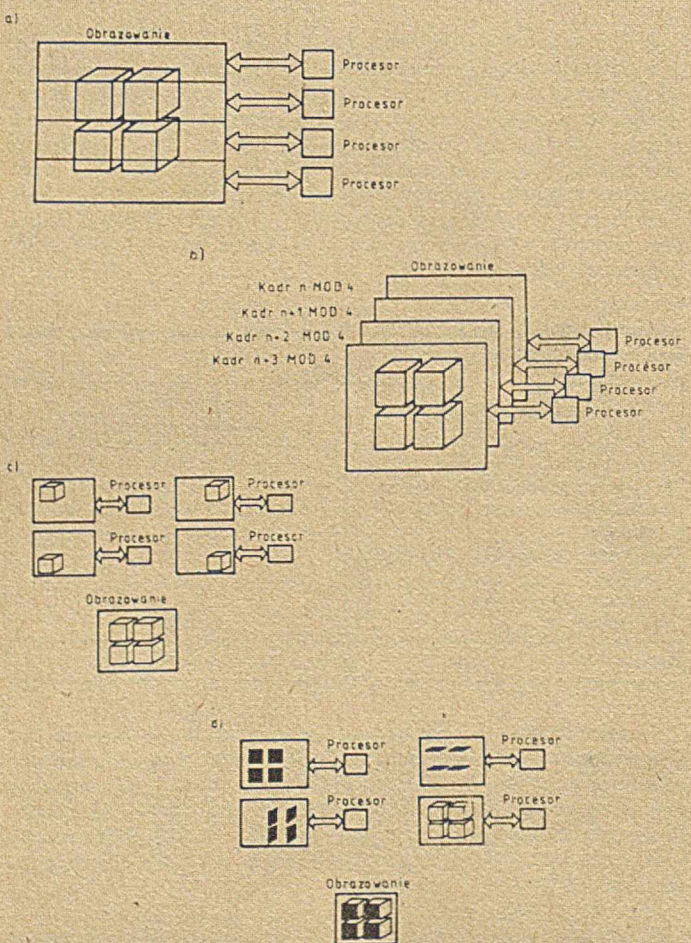
Spełnienie tych wymagań jest możliwe technicznie przy wykorzystaniu techniki transputerowej. Podstawą koncepcji tych systemów jest rozproszenie zadania przetwarzania obrazu i obliczeń.

Możliwości rozproszenia zadania przetwarzania obrazu i obliczeń

Koncepcja zapewnienia teoretycznie dowolnej szybkości przetwarzania obrazu opiera się na podziale zadania przetwarzania na dostatecznie szybkie do obliczenia podzadania i użyciu takiej liczby procesorów, jaka jest wymagana dla ich realizacji. Oczywiście taki podział nie zawsze jest możliwy w dowolnym stopniu, wobec istnienia wzajemnych zależności między podzadaniami.

Podejście takie stwarza dwa podstawowe problemy, z których pierwszy dotyczy sposobu dokonania podziału zadania na podzadania, drugi natomiast – techniki wykonywania obliczeń.

W zakresie rozproszenia zadania przetwarzania obrazu wskazuje się zazwyczaj cztery niezależne metody podziału: podział przestrzenny, chronologiczny, obiektowy i charakterystyczny (rys. 2).



Rys. 2. Metody rozproszenia przetwarzania obrazu: a) przestrzenna, b) chronologiczna, c) obiektowa, d) charakterystyczna

Podział przestrzenny polega na podziale ekranu na fragmenty (bloki) i przyporządkowanie obliczeń związanych z obrazem przedstawianym w danym fragmencie osobnemu procesorowi lub grupie procesorów.

Podział chronologiczny polega na przydzieleniu przetwarzania kolejnych kadrów obrazu oddzielnym procesorom. Na przykład, kolejne fazy obrotu obiektu mogą być obliczane przez oddzielne procesory.

Obiektowa dekompozycja obrazu polega na przydzieleniu poszczególnych obiektów do opracowania oddzielnym procesorom. Położenie obiektu i wartości pikseli, odpowiadających mu na ekranie, są zatem obliczane przez osobny procesor, do którego został on przydzielony.

Teoretycznie najtrudniejszy jest **podział charakterystyczny.** Polega on na przyporządkowaniu poszczególnych cech przetwarzanego obiektu poszczególnym procesorom. Występujące tu trudności dotyczą między innymi zasad i sposobu wyróżniania cech.

Przedstawione teoretyczne możliwości dekompozycji zadania przetwarzania graficznego mogą być w pełni wykorzystane pod warunkiem posiadania modularnych modeli obliczeń dających się łatwo wyrazić w odpowiednim języku programowania oraz oprogramowania systemowego i sprzętu potrzebnych do wykonywania tego zadania.

Obecnie wydaje się, że algorytmy rozproszonego przetwarzania dają się przejrzyście zapisać w języku Occam, a transputery są dobrymi elementami do budowy odpowiednich wieloprocesorów.

Architektura rozproszonego systemu graficznego opartego na transputerach

Architektura rozproszonego systemu do przetwarzania zadań graficznych zostanie omówiona na przykładzie systemu opracowanego przez firmę INMOS. Został on zaprojektowany w ten sposób, aby w jak największym stopniu spełnić poprzednio sformułowane wymagania 1 do 4.

Podstawowym elementem obliczeniowym jest w tym systemie transputer IMS T800. Jest to układ zawierający 32-bitowy procesor o szybkości 10 MIPS (1,2 MFLOP), przy zegarze 20 MHz, zmiennoprzecinkową jednostkę arytmetyczną oraz cztery niezależne łącza szeregowe pracujące z szybkością do 20 Mb/s. Sprzęg z pamięcią zewnętrzną umożliwia adresowanie 4 GB.

Obecnie do programowania transputerów można stosować prawie wszystkie popularne języki programowania, jak C, Pascal, Fortran, jednak pełne możliwości transputerów można wykorzystać programując w języku Occam [3, 5]. Podstawę teoretyczną tego języka stanowi CSP [1]; jest to pierwszy język do programowania równoległego mający szansę stania się standardem.

Zasadniczy wpływ na ukierunkowanie budowy systemów z transputerami wywarła przyjęta przez firmę INMOS koncepcja modułów transputerowych, tzw. TRAM-ów *(Transputer Module)*. TRAM jest to rodzaj sterownika zawierającego transputer, pamięć oraz specjalizowane układy zależne od zastosowania. Wszystkie moduły TRAM mają standardowe:
- sprzęg wykorzystujący wyłącznie łącza szeregowe,
- rozkład wyprowadzeń sygnałów elektronicznych (styków),
- wielkość w sensie rozmiarów fizycznych płytki,

oraz są zaprojektowane zgodnie z opublikowaną specyfikacją.

Tak więc podstawowe sygnały zewnętrzne TRAM-u są całkowicie niezależne od rodzaju transputera (T212, T222, T414, T225, T800, T801, T805 itp.), liczby transputerów w TRAM-ie oraz wielkości pamięci. TRAM stanowi zatem podstawowy blok, umożliwiający bardzo łatwą budowę systemów przez łączenie ze sobą odpowiednich TRAM-ów.

Rodzaje TRAM-ów do zastosowań graficznych

W celu rozproszenia pamięci kadru obrazu oraz zapewnienia odpowiedniej obsługi ekranu opracowano dwa rodzaje TRAM-ów:

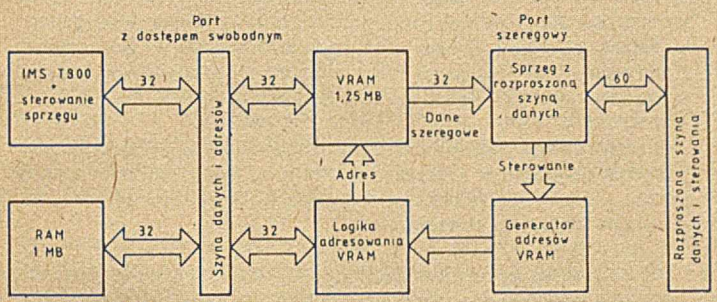
- TRAM z portem szeregowym (serial port TRAM IMS B419). Nazwa pochodzi od trybu przekazywania zawartości pamięci kadru do magis-

trali systemowej, co odbywa się ściśle sekwencyjnie.

- TRAM sterowania monitora (*display backend driver* TRAM IMS B409).
- TRAM z portem szeregowym (rys. 3). Zawiera oprócz transputera T800:

– pamięć o łącznej wielkości 2,25 MB i czasie dostępu 4 cykle (1 cykl równa się 50 ns), z czego 1 MB to pamięć dynamiczna, a 1,25 MB to tzw. pamięć wideo (VRAM), służąca do pamiętania danych opisujących piksle ekranu;

– generator adresów pamięci wideo, generowanych na podstawie sygnałów przekazywanych przez rozproszoną szynę sterowania i danych;



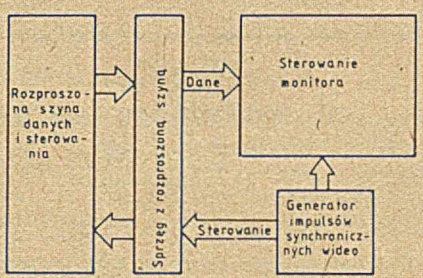
Rys. 3. Schemat blokowy TRAM-u z portem szeregowym

– sprzęg z 60-sygnałową szyną sterowania i danych; szyna ta między innymi doprowadza odpowiednie sygnały taktujące do układów TRAM-u oraz dane przesyłane z VRAM-u do TRAM-u sterowania ekranu. Podstawowymi elementami składowymi tego TRAM-u sterowania monitora są (rys. 4):

– generator sygnałów synchronizujących działanie całego systemu wideo,

– sprzęg z szyną sterowania i danych,

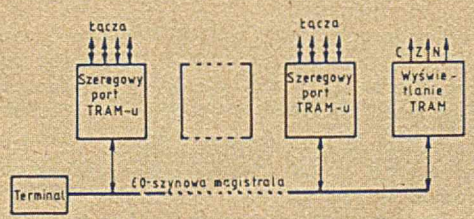
– zależny od zastosowania moduł, którego zadaniem jest sterowanie na podstawie 32-bitowych danych, podawanych przez szynę danych i sterowania zadanego typu monitora.



Rys. 4. Schemat blokowy TRAM-u sterowania monitora

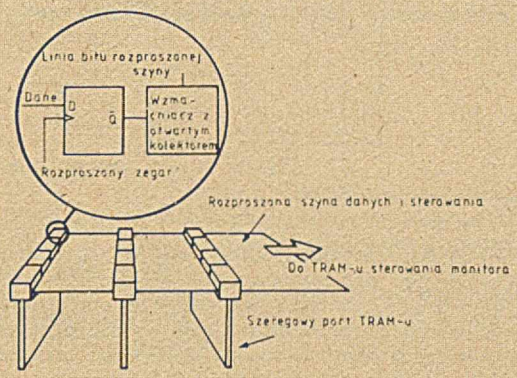
Charakterystyka szyny danych i sterowania

Szyna danych i sterowania jest to 60-szynowa magistrala, spinająca w jeden system TRAM-y z portem szeregowym oraz TRAM sterowania monitora (rys. 5).



Rys. 5. Sposób połączenia TRAM-ów i system

Pod względem elektronicznym dołączenie do szyny stanowi układ z zanegowanym otwartym kolektorem (rys. 6). Umożliwia to sumowanie logiczne na magistrali sygnałów stanowiących dane, podawane z pamięci VRAM poszczególnych TRAM-ów. Usuwa to również konieczność wprowadzenia logiki dostępu do szyny, umożliwiając dowolne rozłożenie pamięci kadru pomiędzy poszczególne TRAM-y z portem szeregowym. Ponieważ wszystkie TRAM-y są zsynchronizowane, zatem każdy z nich może zmienić niezależnie wybraną wartość piksla, która znajduje się na magistrali. Jeżeli żaden z TRAM-ów nie podaje na magistralę wartości opisującej dany piksel, to wartość ta jest równa 0. Podane właściwości szyny danych i sterowania są podstawą rozproszenia przestrzennego, chronologicznego i obiektowego przetwarzania obrazu. Należy podkreślić, że dzięki odpowiedniej konstrukcji systemu rozproszenie to uzyskuje się w sposób zupełnie naturalny.



Rys. 6. Schemat sprzęgu między rozproszoną szyną systemu a TRAM-ami

Pamięć wideo i pamięć kadru

Istotnym zagadnieniem z punktu widzenia rozproszenia pamięci kadru obrazu jest sposób odwzorowania przestrzeni adresowej pamięci wideo w przestrzeń pamięci kadru obrazu. Pamięć wideo adresuje się zazwyczaj przez współrzędne (wiersz, kolumna). Pozostaje zatem kwestia, czy wiersze i kolumny w tej pamięci powinny odpowiadać wprost współrzędnym ułożenia danych w pamięci kadru opisującej wartości poszczególnych piksli na ekranie. Zwykle przyjmuje się jedno z dwu rozwiązań.

W pierwszym z nich dane z tego samego wiersza pamięci wideo są umieszczane w jednym wierszu pamięci kadru. Tak więc zachowuje się prostą relację między wierszami i kolumnami w obu pamięciach. Powoduje to jednak, że wiersz pamięci kadru nie może być dłuższy niż wiersz pamięci wideo, co jest równoważne ograniczeniu liczby piksli w wierszu ekranu. Określając obrazowo ten sposób odwzorowania adresów obu pamięci można stwierdzić, że prostokątny fragment przestrzeni adresowej z pamięci wideo wpisuje się do pamięci kadru z zachowaniem współrzędnych (wiersz, kolumna).

W rozwiązaniu drugim rezygnuje się z powiązania między adresami w pamięci wideo oraz pamięci kadru. Ciągły obszar pamięci wideo jest przesyłany do pamięci kadru, wiersz po wierszu. Wiersze te ulegają załamaniu wypełniając kolejno cały obszar kadru. Ten sposób adresowania nie narzuca ograniczeń na długość wiersza pamięci kadru, wymaga jednak transmisji danych z pamięci wideo w czasie rzeczywistym.

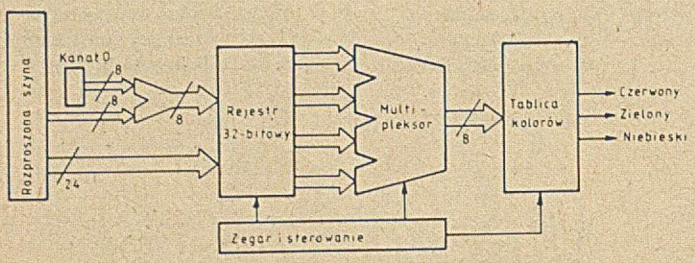
Analizując obydwie metody należy stwierdzić, że druga z nich jest znacznie ogólniejsza, lecz trudniejsza do zrealizowania. W systemie opracowanym przez firmę INMOS wybrano drugą z tych metod jako podstawową. Metoda pierwsza jest dostępna jako opcja przy ograniczeniu liczby piksli w wierszu do 1024 × 8 bitów. Uzyskano w ten sposób niezależność systemu przetwarzania obrazu od rodzaju monitora.

Sposób zmiany rozdzielczości i liczby kolorów

Pozostaje jeszcze do rozważenia sposób przystosowania systemu do pracy z różną rozdzielczością i liczbą kolorów. Odpowiednią elastyczność rozwiązania sprzętowego zapewnia się przez umieszczenie w TRAM-ie sterowania ekranu trzech identycznych niezależnych kanałów sterowania kolorami piksli (rys. 7). Liczba bajtów przetwarzanych przez kanał oraz liczba wykorzystywanych kanałów zależy od trybu pracy. Obecnie opracowane systemy można podzielić pod tym względem na:

- **Systemy 8-bitowe**. Podawane przez magistralę 32-bitowe dane są traktowane jako wartości czterech piksli. W trybie tym wykorzystuje się tylko jeden kanał.

● **Systemy 24-bitowej niskiej rozdzielczości.** W trybie tym 32-bitowe dane z magistrali są traktowane jako wartość jednego piksla, przy czym najbardziej znaczący bajt nie jest używany. Trzy z pozostałych bajtów są rozdzielane między poszczególne kanały, z których każdy jest odpowiedzialny za sterowanie innym kolorem podstawowym.



Rys. 7. Kanał sterowania kolorami piksli

● **Systemy 24-bitowe wysokiej rozdzielczości.** W systemach tych każdy z trzech kanałów pracuje analogicznie jak w systemie 8-bitowym, z tą różnicą, że co trzecie 32-bitowe słowo opisuje ten sam kolor podstawowy i jest podawane do oddzielnego kanału sterującego tym kolorem. Przyjęte rozwiązanie umożliwia zmianę kolorów i rozdzielczości obrazu odpowiednio dla różnych zastosowań.

Konfiguracje systemu do przetwarzania zadań graficznych

Sposób konfiguracji systemu zależy od tego zastosowania, a jego wybór jest zdeterminowany głównie przez liczbę obliczeń potrzebną do wyznaczenia wartości piksli w pamięci kadru. Rozważa się następujące trzy konfiguracje pracy TRAM-u z portem szeregowym:

● Praca jako generator danych. TRAM otrzymuje polecenia graficzne wysokiego poziomu oraz realizuje je przez wykonanie określonych obliczeń i generowanie danych w pamięci kadru.

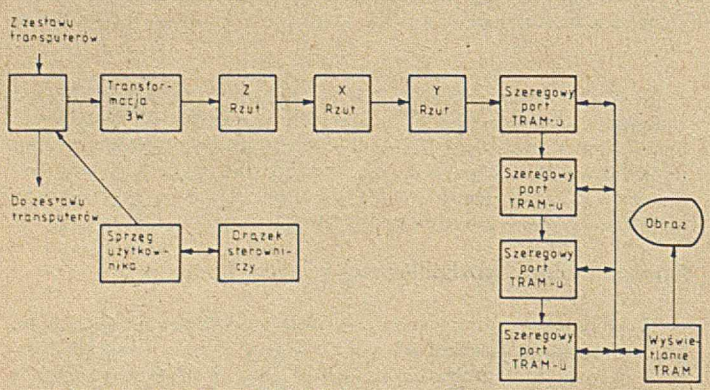
● Praca jako spływ danych. TRAM nie wykonuje w tej konfiguracji żadnych poleceń graficznych, a jego funkcje ograniczają się do zbierania danych napływających do niego przez łącza szeregowe od innych TRAM-ów i umieszczania ich bezpośrednio w pamięci kadru.

● Połączenie obu uprzednio wymienionych konfiguracji.

Jakość działania systemu można poprawić przez zwiększenie liczby TRAM-ów z portem szeregowym i rozproszenie między nie zadania aktualizacji pamięci kadru.

Przedstawiona koncepcja budowy sytemu do przetwarzania zadań graficznych została opracowana z myślą o wykonywaniu następujących zadań:

- kreślenie obrazu zbioru Mandelbroata [4],
- konstruowanie obrazu metodą śledzenia przebiegu promieni świetlnych [2],
- symulowanie lotu [6].



Rys. 8. Konfiguracja systemu symulatora lotu

Dwa pierwsze przykłady są zadaniami wymagającymi bardzo dużej liczby obliczeń. Do ich wykonania stosuje się zazwyczaj zestawy transputerów połączonych w pierścień. Zadanie symulowania lotu wymaga bardzo dużej liczby obliczeń, wynikającej z przetwarzania w czasie rzeczywistym trójwymiarowej sceny, oraz jej wizualizacji, ze względu na konieczność zachowania dynamiki ruchu i jakości obrazu. Konfigurację części systemu związanej z przetwarzaniem graficznym dla tego zastosowania pokazano na rys. 8. Bloki prostokątne reprezentują transputery, którym przyporządkowano określone fazy przetwarzania

* * *

Opisany system do przetwarzania zadań graficznych spełnia w bardzo dużym stopniu przedstawione na początku wymagania. Cechy tego systemu są zdeterminowane przez modularność i bardzo dobre parametry systemów transputerowych. Ze względu na łatwość dostosowania systemu do różnych zadań i jego parametry, stanowi on idealne rozwiązanie dla ośrodków naukowo-badawczych. Koszty tego systemu są jednak znaczne i dlatego celowe jest traktowanie go jako modelu fizycznego, umożliwiającego dalsze przejście do specjalizowanych rozwiązań sprzętowych, zwłaszcza w wypadku zastosowań masowych.

Należy również podkreślić, że naukowe zastosowania grafiki komputerowej prawie zawsze wymagają bardzo dużej liczby obliczeń. Opisany system graficzny zazwyczaj musi współpracować z odpowiednio rozbudowanym systemem obliczeniowym.

Wydaje się, że jest możliwe znaczne zwiększenie mocy obliczeniowej systemów transputerowych, nie tylko przez rozbudowę konfiguracji, lecz również dzięki opracowaniu nowego transputera H1. Jeżeli prace nad tym transputerem powiodą się, to w roku 1991 będzie dostępny nowy transputer o szybkości 20 MFLOPS i 100 MIPS, w pełni zgodny z istniejącymi konstrukcjami.

LITERATURA

[1] Hoare C. A. R.: Communicating Sequential Processes. Prentice-Hall, 1985
[2] High Performance Graphics with the IMS T800. Technical Note, No. 37, INMOS Limited
[3] IMS T800 Architecture. Technical Note, No. 6, INMOS Limited
[4] Notes on Graphics Support and Performance Improvements on the IMS T800. Technical Note, No. 26, INMOS Limited
[5] Occam 2 Reference Manual. INMOS Limited, Prentice-Hall, 1988
[6] A Transputer Based Multi-User Flight Simulator. Technical Note, No. 36, INMOS Limited.

Fizyczne struktury systemów baz danych (2)
Techniki mieszania

dokończenie ze s. 21

Po podstawieniu za g wartości 2 wzór ten staje się identyczny z wzorem określającym spiralną funkcję mieszającą zwykłego mieszania liniowego. Dla wartości g mniejszych niż 2 metoda spiralna uogólnionego mieszania liniowego charakteryzuje się bardziej równomiernym rozkładem rekordów w pliku i redukuje liczbę rekordów nadmiarowych. Odbywa się to kosztem zwiększonej liczby operacji dostępu przy wykonywaniu operacji rozdzielania. Dla zbyt małych wartości g koszt operacji rozdzielania przekracza korzyści wynikające z rozdzielania większej liczby komór jednocześnie. Wybór odpowiedniej wartości współczynnika g jest więc bardzo ważnym czynnikiem wpływającym na efektywność implementacji techniki uogólnionego mieszania liniowego.

LITERATURA

[1] Gardarin G., Valduriez P.: Bases de donnes relationnelles, analyse et comparison des systemes. Edition Eyrolle, 1985
[2] Jagielski R.: Tablice rozproszone. PWN, Warszawa 1982
[3] Kjellberg P., Zahle T. U.: Cascade Hashing. Proc. of the 10-th Int. Conf. on Very Large Data Bases, Singapore, pp. 481–492, August 1984
[4] Knuth D.: The Art of Computer Programming. Vol. 3: Sorting and Searching. Addison-Wesley, Reading (MA) 1973
[5] Larson P.: Linear Hashing with Partial Expansion. Proc. of the 8th Int. Conf. on Very Large Data Bases, Mexico City, pp. 300–309, September 1982
[6] Litwin W.: Linear Hashing: A New Tool for File and Table Addressing. Proc. of the 6th Int. Conf. On Very Large Data Bases, Montreal, pp. 212–223, October 1980
[7] Ramamohanarao K., Lloyd J. W.: Dynamic Hashing Schemes. The Computer Journal, Vol. 25, No., 4, pp. 478–485, 1982
[8] Sibuja M., Jamamoto T.: Algoritmy obrabotki dannych (tłum. z jęz. japońskiego). MIR, Moskwa 1986.

ZBIGNIEW CYLKOWSKI
Instytut Materiałoznawstwa
i Mechaniki Technicznej
Politechnika Wrocławska

Pomiar czasu obliczeń na IBM PC

Programista zazwyczaj chce pisać swoje programy tak, żeby były optymalne m.in. pod względem czasu obliczeń. Aby to realizować, powinien umieć mierzyć czas wykonania napisanych algorytmów, jak również powinien znać – przynajmniej z grubsza – czasy działania poszczególnych instrukcji danego języka programowania.

Niniejszy artykuł ma na celu pomóc w tej sprawie, zakładając, że programista używa mikrokomputera IBM PC i pisze w Turbo Pascalu. Zaproponowano pewną metodę pomiaru czasu, a następnie podano czasy działania wybranych, często występujących w obliczeniach numerycznych, instrukcji języka Turbo Pascal wersja 4.0 (TP4) i Turbo Pascal wersja 5.0 (TP5).

Pomiar czasu na IBM PC

W systemie DOS, jak wiadomo istnieje tzw. licznik zegara. Jest to pewna porcja bajtów pamięci, których zawartość jest uaktualniana co ok. 0,055 s. Ich stan można odczytać w Turbo Pascalu 4.0 i 5.0 za pomocą procedury *getTime* (również w innych językach istnieją możliwości odczytywania stanu licznika, np. w Turbo Pascalu 3.0 można postąpić tak, jak to zrobiono w [2]). Instrukcja *getTime* (*h, m, s, ss*) podaje bieżącą godzinę *h*, minutę *m*, sekundę *s* i setną sekundy *ss*. W szczególności fragment programu:

```
getTime(h0,m0,s0,ss0);
....
getTime(h,m,s,ss);
t:=((h-h0)*60+m-m0)*60+s-s0+0.01*(100+ss-ss0)-1
```

oblicza czas *t* (w sekundach) zużyty na wykonanie instrukcji zawartych między dwoma kolejnymi wywołaniami procedury *getTime*. W podanym fragmencie zmienna *t* jest typu *real* (lub innego typu rzeczywistego), natomiast pozostałe zmienne są typu *word*; dodanie liczby 100 do różnicy *ss-ss0* zabezpiecza przed otrzymaniem ujemnej wartości wyrażenia typu *word*.

Błąd dokonanego w powyższy sposób pomiaru sięga wartości ok. 0,06 s. Jest to niekiedy zbyt dużo, zwłaszcza dla obliczeń trwających ok. 0,1 s. lub krócej. Okazuje się, że istnieje sposób dokładniejszy. Stosując mianowicie (i modyfikując) pomysł zamieszczony w publikacji [1], s. 62–63, można błąd pomiaru zmniejszyć kilkadziesięciokrotnie.

Przed podaniem tego logarytmu rozważmy fragment programu.

```
      getTime(h,h,h,ss);
(*)   j:=0;  repeat  j:=j+1;  getTime(g,g,g,ss0)  until ss0<>ss
```

Spostrzegamy, że wartość zmiennej *j* (typu *integer*) wyznacza jak gdyby czas, jaki upłynął od momentu rozpoczęcia tych obliczeń do momentu zmiany stanu licznika zegara. A gdy te obliczenia zostaną wykonane tuż po zmianie stanu licznika, to *j* wyznaczy czas, jaki upłynie między dwoma kolejnymi uaktualnieniami licznika. Obydwa te czasy mierzy się w jednostkach odpowiadających czasowi wykonania jednego cyklu instrukcji **repeat.**

Prócz tego, wykonując wielokrotnie (kilkanaście tysięcy razy lub więcej) instrukcje {*} (wraz z instrukcją *ss*:=*ss*0 i instrukcjami zapamiętującymi wyniki), zaobserwowano następujące dwa fakty:
1. Ostatni parametr procedury *getTime* przyrastał o 5 lub 6, przy czym przyrosty układały się następująco:

```
5, 6, 5, 6, ..., 6, 5,  5, 6, 5, 6, ..., 6, 5,  5, 6, ...
└── 67 liczb ──┘ └── 67 liczb ──┘
```

Daje to średni przyrost $\frac{368}{67} = 5{,}49253...$

Uwaga: dla niektórych typów komputerów ciąg przyrostów bywa inny, np. 6, 5, 5, 6, 5, 5, ...; w dalszych rozważaniach należy wtedy odpowiednio zmienić średnią.
2. Końcowe wartości zmiennej *j* były niemal stałe: w zależności od komputera, kompilatora i tzw. nakładek na system DOS było np. $21 \leqslant j \leqslant 22$ lub $140 \leqslant j \leqslant 141$.

Obserwacje te pozwalają sądzić, że licznik zegara „chodzi" regularnie, zwiększając swój stan o 0,0549253... co 0,0549253... s (lub wykonując czynności równoważne), natomiast procedura *getTime* zaokrągla odczytaną wartość do setnych części sekundy.

Zapowiedziany algorytm jest następujący (mierzy on czas instrukcji zaznaczonych kropkami):

```
      getTime(h,h,h,ss);
      repeat  getTime(h0,m0,s0,ss0)  until  ss0<>ss;
      ....
      getTime(h,m,s,ss);
      t:=2;  repeat  t:=t+1;  getTime(g,g,g,st);  until  st<>ss;
(**)  j:=0;  repeat  j:=j+1;  getTime(g,g,g,sj);  until  sj<>st;
      t:=((h-h0)*60+m-m0)*60+s-s0+0.01*(100+ss-ss0)-1;
      t:=(int(t/0.0549253+1.5)-t/j)*0.0549253
```

Zakłada się tu, że $h \geqslant h0$, innymi słowy oznacza to, że podczas pomiaru czasu nie może minąć północ.

Sens ostatniej instrukcji algorytmu stanie się jeśniejszy, jeśli zauważymy, że użyte w niej wyrażenie ma wartość

$$\left(\left[\frac{t}{0{,}0549253}+0{,}5\right]+\frac{j-i}{j}\right)*0{,}0549253$$

i że część całkowita $[u+0{,}5]$ jest zaokrągleniem wartości u do najbliższej liczby całkowitej. W naszym przypadku ta część całkowita jest liczbą cyknięć zegara.

Początkowa wartość zmiennej *i* (przypisanie *i*:=2) została dobrana tak, aby czas wykonywania instrukcji pustej był najbardziej zbliżony do zera. Tę wartość należy oczywiście dostosować do używanego komputera – jej zmiana o 1 powoduje, iż końcowe *t* zostaje zmienione o ok. $0{,}055/j$, co widać z ostatniej instrukcji algorytmu.

Należy spodziewać się, że błąd pomiaru czasu wynosi teraz $\frac{0{,}005}{j}$ *s*. Jest interesujące, że w tej metodzie błąd pomiaru nie uległby zmianie, gdyby licznik zegara „chodził" nawet rzadziej, wtedy bowiem wartość *j* obliczona w instrukcjach {**} byłaby proporcjonalnie większa.

Czasy wykonywania poszczególnych instrukcji

Poniżej poznamy czasy działania wielu prostych instrukcji pascalowych. Za ich pomocą można dość łatwo przewidywać z grubsza czas wykonywania instrukcji nieco bardziej skomplikowanych, co miejmy nadzieję, pozwoli pisać szybkie wersje programów. Dodajmy, że czasy mierzono z dokładnością do ok. 3 cyfr znaczących, co uzyskano przez powtarzanie wybranej instrukcji dostateczną liczbę razy oraz wyeliminowanie czasu zużytego na organizację powtórzeń.

Mierząc czas zauważono, że zależy on na ogół od tego czy adres argumentów danej operacji jest parzysty, czy nieparzysty. Np. instrukcja $k:=i+1$ w programie

var b:byte; i, k: integer; begin ... k:=i+1; ...end.

jest wykonywana wolniej niż w programie

var i, k integer; b: byte; begin ... k:=i+1; ...end.

Zmienne *i, k* mają w pierwszym programie adresy nieparzyste, a w drugim – parzyste. Pochodzące z takich powodów względne zmiany czasu obliczeń są znikomo małe dla instrukcji wykonywanych długo, ale dla instrukcji szybkich mogą wynosić ok. 50%.

Zauważono również, że czas może zależeć od parzystości adresu pierwszego bajtu danej instrukcji (w programie wynikowym). Jeśli np. w poprzednim przykładzie umieścimy instrukcję *write* ('...') przed instrukcją $k:=i+1$, to czas działania tej drugiej zależy od parzystości liczby znaków drukowanych przez instrukcję pierwszą. W tabelach 1, 2 i 3 podano czasy średnie pod tym względem.

Pomiarów czasu dokonywano na mikrokomputerze IBM PC/XT wyposażonym w procesor 8088, koprocesor 8087 i zegar o częstotliwości ok. 8 MHz oraz na IBM PC/AT z procesorem 80286, koprocesorem 80287 i zegarem o częstotliwości także ok. 8 MHz.

Wybrane instrukcje podzielono na dwie grupy:
a) instrukcje nie używające żadnej wielkości typu rzeczywistego,
b) instrukcje, w których występują wielkości typu rzeczywistego.

Instrukcje z grupy b były wykonywane w dwu wersjach:
- z użyciem koprocesora, tj. z dyrektywą {$n+},
- bez użycia koprocesora, tj. z dyrektywą {$n−}.

Wszystkie instrukcje wykonano z dyrektywą {$r−}, czyli bez sprawdzania czy wskaźniki, wielkości całkowite itp. mieszczą się w odpowiednich granicach.

Dla tabeli 1 (grupa a) definicje typów, deklaracje i wartości zmiennych są następujące:

```
type
   IT = integer;      {lub longint}
var
   i,j,k,m:          IT;
   ita:              array[1..10] of IT;
   c,c0:             char;
   ip,jp,kp:         ^IT;
   z,z1:             set of 1..40;
   rec:              record  m: IT;  a: real  end;
   b,b1,b2:          Boolean;
procedure proc0(n: IT);  begin end;

i:=3;    j:=5;    m:=10;
c0:='a';
ip^:=3;    jp^:=5;
z:=[2,3,5,7,11,13,17,19,23,29];    z1:=[5,3,23,15,11];
b1:=true;    b2:=false;
```

Tabela 1 zawiera czasy działań obliczone w TP4. Czasy działań w TP5 są niemal identyczne, z wyjątkiem niektórych działań dotyczących stałych typu **string** i **set**. Należy w szczególności zwrócić uwagę na to, że:
1. Instrukcje

$b := 'jajo' < 'jajko';$ $b := 'jajo' < 'kura'$

są w TP5 bardzo szybkie, zajmują tyle czasu co przypisanie zmiennej *b* odpowiedniej stałej boolowskiej (zauważono też, że porównywanie zmiennych łańcuchowych jest w TP5 odrobinę szybsze niż w TP4).
2. Instrukcja

$b := j$ **in** [2,3,5,7,11,13,17,19,23,29]

jest ok. 10 razy szybsza w TP5 niż w TP4.
3. Instrukcja

$b := [5,3,23,15,11] <= z$

jest ok. 2,5 razy szybsza w TP5 niż w TP4.

Tabela 1. Instrukcje z grupy a; czas w μs

	PC/XT		PC/AT	
IT =	integer	longint	integer	longint
for k:=1 to m do	137.0	268.0	84.0	158.0
for k:=1 to 10 do	110.0	212.0	69.0	128.0
k:=i	5.5	12.0	2.7	6.1
k:=-i	6.5	17.0	3.2	8.7
k:=i+j	8.9	19.0	4.2	9.4
k:=i+5	7.0	15.0	3.5	7.1
k:=k+1	6.0	15.0	3.0	7.1
k:=i*j	24.0	87.0	6.2	34.0
k:=m div j	29.0	479.0	7.1	289.0
k:=m mod j	29.0	482.0	7.1	294.0
k:=m shl 5	9.9	50.0	4.0	32.0
k:=m shl 15	15.0	93.0	5.3	62.0
k:=not i	6.5	14.0	3.2	7.0
k:=i or j	8.9	18.0	4.2	9.4
k:=i xor j	8.9	18.0	4.2	9.4
k:=i and j	8.9	18.0	4.2	9.4
k:=sqr(m)	21.0	83.0	4.3	32.0
k:=abs(m)	8.8	26.0	5.5	17.0
k:=succ(m)	6.0	15.0	3.0	7.1
k:=pred(m)	6.0	15.0	3.0	7.1
inc(k)	4.7	9.9	2.1	5.5
inc(k,j)	7.4	15.0	4.0	7.1
inc(k,5)	4.9	9.9	2.8	5.5
inc(ita[i])	9.1	15.0	4.9	8.0
ita[i]:=ita[i]+1	16.0	27.0	7.6	14.0
k:=random(m)	103.0	107.0	45.0	47.0
proc0(i)	36.0	41.0	22.0	24.0
c:=succ(c0)	5.5	5.5	3.2	3.2
k:=ord(c0)	6.0	10.5	3.2	5.4
b:=i<j	12.0	21.0	7.1	12.0
if i<j then	7.0	16.0	3.3	7.1
b:=odd(j)	8.9	8.9	6.2	6.2
b:=b1	4.5	4.5	2.7	2.7
b:=not b1	9.9	9.9	7.1	7.1
b:=b1 or b2	9.1	9.1	6.2	6.2
b:=b1 xor b2	7.4	7.4	4.2	4.2
b:=b1 and b2	14.5	14.5	9.2	9.2
b:='jajo' < 'jajko'	68.0	68.0	44.0	44.0
b:='jajo' < 'kura'	59.0	59.0	37.0	37.0
b:=j in z	141.0	141.0	77.0	77.0
b:=j in [2,3,5,7,11, 13,17,19,23,29]	475.0	475.0	262.0	262.0
b:=z1 <= z	237.0	237.0	128.0	128.0
b:=[5,3,23,15,11] <= z	384.0	384.0	207.0	207.0
kp:=ip	12.0	12.0	6.1	6.1
b:=kp=ip	19.0	19.0	11.0	11.0
kp^:=i	9.9	18.0	5.2	8.1
kp^:=ip^	15.0	20.0	7.7	11.0
kp^:=ip^+jp^	22.0	31.0	12.0	17.0
rec.m:=i	5.5	12.0	2.7	6.1
rec.m:=rec.m+i	8.9	19.0	4.2	9.4

Dla tabel 2 i 3 (instrukcje z grupy *b*) definicje typów, deklaracje i wartości zmiennych są następujące:

```
type
    RT = real;          {lub single, lub double, lub extended}
   TAB = array[1..10] of RT;
var
    i,j:            integer;
    p,q,r,x:        RT;
    a:              TAB;
    aa:             array[1..10] of TAB;
    rp:             ^RT;
    rec:            record  m: integer;  a: RT end;
procedure proc1(x: RT);  begin end;
procedure proc2(x,y: RT);  begin end;
procedure proc3(x: RT; var y: RT);  begin end;
procedure proc4(x: RT; var y: RT);  begin y:=x end;

  i:=3;       j:=5;
  p:=3.21;    q:=-12.34;
```

W tabelach 2 i 3 czasy wyznaczania wartości funkcji

int(x), frac(x), round(x), trunc(x), abs(x), sqrt(x), sin(x), cos(x), arctan(x), exp(x), ln(x)

są średnimi obliczonymi dla kilkunastu wybranych wartości *x*, odmiennych dla różnych funkcji.

Tabela 2 zawiera czasy działań wykonywanych z wykorzystaniem koprocesora arytmetycznego. Spostrzegamy znaczne różnice między czasami działań na typach *real* i *extended*. Wyjaśnienie tego faktu nie jest trudne: wielkości tych typów są pamiętane w odmienny sposób, a wszystkie działania zmiennopozycyjne są wykonywane w typie

Tabela 2. Instrukcje z grupy b; dyrektywa {$n+1}, czas w μs

RT =	PC/XT real TP4	PC/XT real TP5	PC/XT extended TP4	PC/XT extended TP5	PC/AT real TP4	PC/AT real TP5	PC/AT extended TP4	PC/AT extended TP5
r:=p	19	19	20	20	10	10	42	42
r:=1.23	70	13	21	20	66	7	44	44
r:=j	81	81	17	18	52	52	37	38
r:=5	13	13	21	21	7	7	44	43
r:=-p	122	115	22	23	102	98	49	50
r:=-1.23	71	13	23	21	76	7	51	43
r:=p+q	332	168	43	43	259	163	92	92
r:=p+1.23	131	123	41	42	133	130	91	90
r:=1.23+p	287	124	42	42	216	130	91	91
r:=p+j	126	119	38	39	128	123	86	86
r:=p*q	796	173	48	48	583	172	103	103
r:=p*1.23	136	131	48	49	147	144	106	105
r:=1.23*p	754	125	47	49	556	144	105	106
r:=p*j	123	120	40	40	127	125	89	89
r:=p/q	926	176	55	55	673	188	118	118
r:=p/1.23	145	133	55	55	160	157	118	119
r:=1.23/p	886	134	56	56	657	166	120	120
r:=p/j	138	128	52	53	154	150	114	113
r:=j/p	894	133	54	54	627	163	115	116
r:=i/j	89	89	51	51	131	132	111	111
if p<q then	257	120	36	35	211	113	66	67
if p<1.23 then	87	80	37	38	91	88	69	69
if p<j then	79	73	32	33	82	78	63	63
r:=sqr(p)	132	125	42	42	142	137	90	90
r:=random	157	158	108	109	153	153	124	124
r:=int(x)	158	143	58	51	143	136	81	80
r:=frac(x)	173	159	73	66	178	173	117	114
i:=round(x)	99	84	48	41	95	85	61	57
i:=trunc(x)	113	98	62	54	112	103	76	73
r:=abs(x)	122	115	22	23	103	98	49	49
r:=sqrt(x)	185	132	85	47	180	143	120	91
r:=sin(x)	381	398	280	306	593	579	532	527
r:=cos(x)	409	400	309	308	654	580	593	528
r:=arctan(x)	302	328	203	235	411	426	349	370
r:=exp(x)	395	436	294	345	597	676	535	620
r:=ln(x)	291	311	191	227	394	424	333	372
a[i]:=p	27	27	39	40	14	14	42	42
aa[i,j]:=p	59	58	60	60	25	25	48	48
aa[i]:=a	192	192	279	279	95	95	139	139
a[3]:=p	19	19	20	21	10	10	42	42
aa[3,5]:=p	19	19	20	20	10	10	42	42
proc1(p)	45	53	51	62	27	28	60	33
proc2(p,q)	59	67	71	84	34	35	99	44
proc3(p,q)	50	58	58	67	29	31	61	36
proc4(p,r)	72	80	81	90	41	43	104	78
rec.a:=p	19	19	20	20	10	10	42	42
rec.a:=1.23	70	13	20	21	66	7	44	44
rp^:=p	23	23	23	24	12	12	42	42
rp^:=1.23	74	17	24	24	69	9	44	44
r:=1.23*p+3.21*q	1811	226	101	100	1359	276	217	217
r:=p*1.23+q*3.21	239	216	101	101	282	275	218	218

Tabela 3. Instrukcje z grupy b; dyrektywa {$n−}, czas w μs

RT =	PC/XT real TP4	PC/AT real TP4	PC/AT real TP5
r:=p	19	10	10
r:=1.23	13	7	7
r:=j	81	52	51
r:=5	13	7	7
r:=-p	23	11	11
r:=-1.23	13	7	7
r:=p+q	121	79	78
r:=p+1.23	103	64	63
r:=p+j	169	108	107
r:=p*q	587	403	378
r:=p*1.23	587	404	377
r:=p*5.0	252	173	164
r:=5.0*p	582	400	374
r:=p*j	651	446	418
r:=p/q	720	491	474
r:=p/1.23	747	531	506
r:=1.23/p	725	505	483
r:=p/j	836	602	567
r:=j/p	778	536	514
r:=i/j	881	612	591
if p<q then	49	30	31
if p<1.23 then	48	33	33
if p<j then	114	77	75
r:=sqr(p)	583	401	374
r:=random	94	49	49
r:=int(x)	78	47	47
r:=frac(x)	227	145	142
i:=round(x)	89	56	59
i:=trunc(x)	88	56	59
r:=abs(x)	20	10	10
r:=sqrt(x)	4371	3093	2947
r:=sin(x)	6855	4718	4431
r:=cos(x)	7096	4881	4593
r:=arctan(x)	6105	4189	3964
r:=exp(x)	6190	4251	4018
r:=ln(x)	7350	5028	4732
a[i]:=p	27	14	14
aa[i,j]:=p	59	25	25
aa[i]:=a	192	95	96
a[3]:=p	19	10	10
aa[3,5]:=p	19	10	10
proc1(p)	45	27	28
proc2(p,q)	59	34	35
proc3(p,q)	50	29	31
proc4(p,r)	72	41	43
rec.a:=p	19	10	10
rec.a:=1.23	13	7	7
rp^:=p	23	12	12
rp^:=1.23	18	9	9
r:=1.23*p+3.21*q	1280	879	827
r:=p*1.23+q*3.21	1281	880	827

extended; działania na typach *real* są zatem okupione stosunkowo kosztowną zmianą typu. W przypadku RT = *double* czy RT = *single* czasy działań były zbliżone do otrzymanych przy RT = *extended* z tym, że dla typu *double* były one często trochę większe, a dla typu *single* często trochę mniejsze niż dla typu *extended*.

Tabela 3 zawiera czasy działania instrukcji z grupy b, wykonywanych przy wyłączonym – za pomocą dyrektywy {$n−} – koprocesorze arytmetycznym. Dla mikrokomputera PC/XT podano czasy obliczone tylko w TP4, gdyż w TP5 były one prawie takie same (jedynie wywołania procedur *proc*1, *proc*2, *proc*3 i *proc*4 były o 8 μs dłuższe.

Chciałbym przy okazji bardzo serdecznie podziękować doc. dr. hab. Bogdanowi Rompoltowi, z Uniwersytetu Wrocławskiego, za wspaniałomyślne użyczenie mi własnego komputera oraz komputerów zainstalowanych w Instytucie Astronomii. Dzięki temu podane tabele czasów można było uzupełnić o dane dotyczące PC/XT. Równie serdecznie dziękuję dr. Wojciechowi Myszce, z Politechniki Wrocławskiej, który życzliwie odpowiadał na moje pytania, a były to pytania człowieka dość nieobeznanego z mikrokomputerami.

LITERATURA

[1] Jerzykiewicz K., Szczepkowicz J.: ALGOL 1204 – System programowania maszyny cyfrowej ODRA 1204. PWN, Warszawa 1972

[2] Kadlof A.: W pogoni za mikrosekundami. Komputer nr 8, s. 30–32, 1988.

Wystawa i konferencje SMAU '90

Dominacja dwóch największych na kontynencie europejskim corocznych światowych przeglądów najnowszych osiągnięć technologii informatycznej i telekomunikacyjnej, jakimi od wielu już lat są hanowerski CEBIT oraz paryski SICOB, sprawia, że bardzo mało znane w Polsce są inne imprezy europejskie o analogicznej tematyce ekspozycji i równie długoletniej międzynarodowej tradycji. Do imprez takich należy zaliczyć SMAU, która jest organizowana na terenie Targów Mediolańskich. Od 4 do 8 października br. SMAU odbędzie się już po raz dwudziesty siódmy. Według stanu zgłoszeń z połowy maja br. w imprezie uczestniczyć będzie 830 wystawców, reprezentujących ok. 2000 firm, z czego ponad połowa spoza Włoch. Powierzchnia wystawowa wyniesie ok. 150 000 m² (10% wzrostu w stosunku do roku ubiegłego) i tym samym zbliży się do rozmiarów wystawy SICOB.

Zgodnie z praktyką podobnych imprez, również w czasie SMAU odbywają się konferencje naukowe o charakterze krajowym lub międzynarodowym. Podczas imprezy tegorocznej tego rodzaju konferencji odbędzie się aż 15. Ich współorganizatorami są instytucje publiczne i prywatne, stowarzyszenia zawodowe oraz czołowe wyższe uczelnie, zarówno włoskie jak i zagraniczne. Tematyka tych konferencji dotyczyć będzie między innymi następujących zagadnień:

- stan obecny oraz tendencje rozwoju informatyki i telekomunikacji;
- technologia informatyczna podstawową strategią przy formowaniu modelu przedsiębiorstwa,
- ewolucja technologiczna i rozwój sieci telekomunikacyjnej,
- rozwój technologiczny gwarancją osiągnięcia przewagi rynkowej,
- zastosowanie metod CAD/CAM w projektowaniu,
- czynnik ludzki w przyszłościowych systemach informatycznych,
- ekologia i informatyczne systemy wiedzy w społeczeństwie przyszłości,
- telekomunikacja i giełda pieniężna,
- reforma samorządów terenowych.

Głównym akcentem tegorocznego programu konferencji będzie w dniu 5 października impreza międzynarodowa, ukierunkowana na skutki przemian politycznych i gospodarczych, jakie zaszły ostatnio w Europie Środkowej i Wschodniej, stwarzając warunki szerokiego otwarcia się tych rynków dla firm włoskich. Jej organizatorem są władze samorządowe Lombardii, najbardziej uprzemysłowionego regionu Włoch. Czynny udział w tej konferencji zapowiedzieli zaproszeni ministrowie i reprezentanci nauki ZSRR, Polski, Czechosłowacji i Węgier, a jej centralnym punktem będzie dyskusja panelowa z udziałem przedstawicieli wymienionych krajów.

(WK)

MEDCOM

ZASILANIE AWARYJNE

ZASILACZE BEZPRZERWOWE

naszej produkcji

UPS–200
UPS–300

produkcji tajwańskiej

UPS–500

produkcji japońskiej

GENERATORY PRĄDOTWÓRCZE

różnych mocy

EO/3/90

Biuro
Al. Ujazdowskie 26/39
00–478 Warszawa tel. 28 93 57, teleks 81 70 60

ZAKŁAD PRODUKCYJNO-USŁUGOWY
ul. Wołodyjowskiego 41
02–724 Warszawa tel.43 04 41

PIOTR ŻYCHOWSKI
Instytut Informacji Naukowo-Technicznej
Uniwersytet Śląski
Sosnowiec

Automatyczna standaryzacja słów tekstu w języku polskim

Jednym z zasadniczych problemów przy automatyzacji systemów informacyjnych jest zadanie formowania obrazu wyszukiwawczego dokumentu lub kwerendy. Jedną z trudności występujących w tym procesie jest umiejętność automatycznego określenia przynależności słowa tekstu do ustalonej gramatycznej formy wzorcowej.

W artykule opisano poszukiwanie oraz wyniki stosowania algorytmu realizującego doprowadzanie słów tekstu języka polskiego do ustalonej formy gramatycznej.

DROGA POSZUKIWAŃ

Na wstępie prac przyjęto pewne uproszczenia. Głównym uproszczeniem jest to, że słowa będące formami pochodnymi w sensie deklinacji, koniugacji itp. wyrażają to samo pojęcie, co forma podstawowa (opuszczenie zagadnień synonimii, homonimii itp.). Automatyczne doprowadzanie słów do jednej formy podstawowej ma niekiedy nazwę „doprowadzania do formy kanonicznej". Za formę kanoniczną można przyjąć: temat słowa, gramatyczną formę podstawową lub inne – uznane za odpowiednie – postacie słów. Problem ten, jakkolwiek często rozwiązywany w językach zachodnioeuropejskich, stanowi znaczną przeszkodę w systemach opartych na językach słowiańskich, w których doprowadzanie do postaci kanonicznej realizuje się zazwyczaj za pomocą jednego z dwóch algorytmów:

- analizującego kombinacje użytych w słowach morfemów głównych i pobocznych,
- porównującego tekstowe formy słów ze słownikiem wszystkich możliwych form.

Obydwa algorytmy wiążą się z koniecznością utworzenia, a później przeszukiwania dużych zbiorów form gramatycznych i (lub) ich elementów. W dążeniu do prostoty algorytmu przyjęto, że formą kanoniczną może być tylko postać słowa występująca w analizowanym tekście.

W początkowej wersji algorytmu za formę kanoniczną (a więc nie koniecznie podstawową w sensie gramatycznym)-przyjmowano pierwszą napotkaną w tekście formę słowa. Uzyskiwane listy słów po standaryzacji często wywoływały kontrowersje ze względu na postać słów uznanych za formy kanoniczne. Równocześnie usiłowano odróżniać przedrostki słów. Niestety, to wymaganie zwielokrotniało, średnio trzykrotnie, czas analizy słowa. W dalszych pracach opuszczono więc proces odrzucania przedrostków. Powyższe założenia i wyniki testów tak określonych założeń podano w [4].

Opierając się na wnioskach językoznawczych [2, 3] i teorioinformacyjnych interpretacjach języka naturalnego [1], jako wymaganie dotyczące ostatecznej formy kanonicznej przyjęto, aby była to najkrótsza z form występujących w tekście, oczywiście spełniająca rygory przynależności do jednej formy, w sensie gramatycznym. Tak zmieniony algorytm stosowano dla różnych tekstów, np. naukowo-technicznych, prawnych, tekstów programów komputerowych, kart dokumentacyjnych i innych.

ZASADY DZIAŁANIA ALGORYTMU

Wprowadzono dwa parametry:

- współczynnik korelacji *WK*,
- graniczny współczynnik korelacji *GWK*.

Na podstawie relacji zachodzących między wartościami tych parametrów, stwierdza się stopień identyczności z określoną formą kanoniczną. Parametr *WK* oblicza się jako iloraz liczby kolejnych identycznych pozycji, występujących w jednym ciągu liter wyrazu tekstu i formy kanonicznej, do liczby liter dłuższego z wyrazów. W wersji początkowej formą kanoniczną była pierwsza postać słowa napotkana w tekście i obowiązywała do końca działania algorytmu. W późniejszych realizacjach forma kanoniczna ulega (w trakcie pracy algorytmu) zmianie prowadzącej do wybrania najkrótszego słowa występującego w tekście, spełniającego warunek $WK >= GWK$. Jeśli występowało więcej niż jedno takie słowo, to za formę kanoniczną przyjęto pierwszą z napotkanych w tekście.

Parametr *GWK* jest wartością stałą ustaloną doświadczalnie: stanowi on próg, którego przekroczenie przez *WK* powoduje uznanie słowa tekstu jako równoważnego dla określonej formy kanonicznej. Oczywiście, rozpatruje się słowo tekstu w stosunku do wszystkich wcześniej ustalonych form kanonicznych i o równoważności decyduje maksymalna wielkość *WK*.

Przykładowo, jeżeli w tekście występuje słowo *JĘZYKOWY*, a w zbiorze form kanonicznych znajduje się słowo *JĘZYK*, to *WK* dla tych wyrazów ustala się następująco:

słowo tekstu – *JĘZYKOWY* = 8 znaków
forma kanoniczna – *JĘZYK* = 5 znaków
+ + + + + = ciąg 5 znaków identycznych,

stąd $WK = 5/8$, czyli 0,625.

Ustalenie $GWK = 0{,}625$ umożliwi uznanie obu słów za wyrażające to samo pojęcie. W wersji ulepszonej, gdyby słowo tekstu było krótsze niż forma kanoniczna (i oczywiście $WK >= GWK$), to stałoby się ono nową formą kanoniczną.

W procedurze tej zasadniczą rolę odgrywa wielkość *GWK*. Jej zawyżenie prowadzi do nierozróżniania ewidentnie wspólnych form słów, zaniżenie natomiast powoduje uznawanie za jednoznaczne słów nie mających związku ze sobą. Jeżeli do tego wziąć pod uwagę, iż występują zarówno słowa krótkie (np. 4–5 literowe), jak i słowa długie (np. 13–15 literowe), to dobór *GWK* zawsze powoduje niedokładności wyników porównań w którejś z tych grup. W trakcie korzystania z algorytmu ustalono optymalny poziom *GWK* na 0,62. W doborze tej wielkości i, co jest równoznaczne, w ocenie sposobu przyporządkowania słów formom kanonicznym korzystano z pomocy Ośrodka Poradnictwa Językowego Instytutu Języka Polskiego Uniwersytetu Śląskiego.

Ocenę prowadzono na podstawie dwóch zestawień:

- listy alfabetycznej słów tekstu (wraz z ich częstością) przed standaryzacją,
- listy alfabetycznej słów (również z częstością) po standaryzacji.

Jako kryteria oceny przyjmowano następujące sposoby identyfikacji słów tekstu z formą kanoniczną:

a) całkowicie poprawne,
b) częściowo poprawne (nie wszystkie słowa udało się doprowadzić do jednej formy kanonicznej),
c) złe (błędna identyfikacja słowa i formy kanonicznej),
d) nie wykonane (spośród słów pochodzących z jednej podstawowej formy gramatycznej nie wybrano żadnej formy kanonicznej, a tym samym nie zidentyfikowano związku tych słów).

Nieuznanie równorzędności słów wprowadza co najwyżej nieuzasadnione zwielokrotnienie liczby form kanonicznych, błędne uznanie równorzędności słów może natomiast prowadzić do fałszywych wniosków.

Przy użyciu $GWK = 0{,}62$ powyższe kryteria wykazywały:
a) całkowicie poprawnie 66%,
b) częściowo poprawnie 15%,
c) źle 7%,
d) nie wykonane 12%.
Oczywiście wyniki powyższe stanowią uśrednienie dla ocen różnych tekstów. Proporcje te, a tym samym ustalona wartość GWK, stanowią kompromis między błędami popełnionymi a działaniami nie wykonanymi.

IMPLEMENTACJA ALGORYTMU

Standaryzacja słów tekstu może prowadzić m.in. do dwóch efektów:
- utworzenia tekstu, w którym każde słowo powinno być zastąpione formą kanoniczną,
- utworzenia listy form kanonicznych, np. do potrzeb zliczania ich częstości.

W obu przypadkach danymi wejściowymi jest tekst. Przetwarzanie wiąże się z koniecznością analizowania kolejnych wierszy tekstu i porównywania z tworzonym zbiorem form kanonicznych. Algorytm realizujący ten problem był programowany w językach Cobol, Fortran, Basic i Pascal, programy natomiast uruchamiano na komputerach Odra, Amstrad i IBM PC.

Przetwarzanie tekstu realizowano w trzech etapach. W pierwszym etapie działał program „czyszczący" tekst ze znaków nie będących literami. Drugim etapem było działanie programu sporządzającego listę słów użytych w tekście. Jako ostatni działał program standaryzujący zadaną listę słów. Podział na kroki działania wynika nie z wielkości (objętości) programów, a z czasu, jaki zajmuje wykonanie wymienionych operacji (oczywiście proporcjonalnego do objętości przetwarzanego tekstu).

Ilustrację działania algorytmu przedstawiono w załączonej tabeli zawierającej wyniki przetwarzania tekstu roboczej wersji tego opracowania (lista pierwszych, alfabetycznie, 58 słów tekstu).

* * *

Przedstawiony sposób standaryzacji słów nie jest doskonały pod względem realizacji zasad gramatycznych. Jednak wraz z objętością tekstu zwiększa się prawdopodobieństwo wystąpienia w nim słowa w podstawowej formie gramatycznej. Wtedy, o ile nie jest to wyjątek, stanowi ona formę kanoniczną.

Przedstawiony algorytm zapewnia dość szybkie ustalenie zbioru form kanonicznych, a tym samym ułatwia automatyczne utworzenie reprezentacji zbioru pojęć użytych w tekście.

Działanie algorytmu w programie standaryzującym

Lp.	Przed standaryzacją	Po standaryzacji
1	A	A
2	Abramson	Abramson
3	Aby	Aby
4	Alfabetyczna	- - - - - - - - - -
5	Alfabetyczną	Alfabetyczną
6	Algorytm	Algorytm
7	Algorytmu	- - - - - - - - - -
8	Algorytmów	- - - - - - - - - -
9	Amstrad	Amstard
10	Analizującego	Analizującego
11	Analizy	Analizy
12	Automatyczna	- - - - - - - - - -
13	Automatyczne	Automatyczne
14	Automatycznego	- - - - - - - - - -
15	Automatycznym	- - - - - - - - - -
16	Automatyzacja	Automatyzacja
17	B	B
18	Basic	Basic
19	Bazując	Bazując
20	Był	- - - - - - - - - -
21	Była	- - - - - - - - - -
22	Było	- - - - - - - - - -
23	Być	Być
24	Błędami	Błędami
25	Błędna	- - - - - - - - - -
26	Błędne	Błędne
27	Będące	Będące
28	Będących	- - - - - - - - - -
29	C	C
30	Całkowicie	Całkowicie
31	Centrum	Centrum
32	Ciąg	Ciąg
33	Ciągu	- - - - - - - - - -
34	Co	Co
35	Cobol	Cobol
36	Czas	Czas
37	Czasu	- - - - - - - - - -
38	Czyli	Czyli
39	Czyszczący	Czyszczący
40	Często	Często
41	Częstości	Częstości
42	Częstością	- - - - - - - - - -
43	Częściowo	Częściowo
44	D	D
45	Dalszych	Dalszych
46	Danymi	Danymi
47	Decyduje	Decyduje
48	Deklinacje	Deklinacje
49	Dla	Dla
50	Do	Do
51	Doborze	Doborze
52	Dobór	Dobór
53	Dodatku	Dodatku
54	Dokumentacyjne	Dokumentacyjne
55	Dokumentu	Dokumentu
56	Doprowadzania	- - - - - - - - - -
57	Doprowadzanie	- - - - - - - - - -
58	Doprowadzić	Doprowadzić

LITERATURA

[1] Abramson N.: Teoria informacji i kodowania. Warszawa, PWN 1969
[2] Lyons J.: Wstęp do językoznawstwa. Warszawa, WNT 1977
[3] Sambor J.: Językoznawstwo statystyczne dla pracowników informacji naukowej. Centrum INTE, materiały szkoleniowe nr 1, Warszawa 1978
[4] Żychowski P.: Metoda automatycznego tworzenia postaci kanonicznej słów w języku polskim. Prace Naukowe Uniwersytetu Śląskiego nr 391, pp. 123–127, 1980.

Systemy otwarte i ich wrogowie, czyli co dalej z UNIX-em (1)
Rozłam w UNIX-ie

dokończenie ze s. 29

dobnie pierwszy prawdziwie wieloprocesorowy wariant UNIX-a. Natomiast jeszcze w zeszłym roku powstał jedyny dotąd produkt OSF: OSF/Motif – rozbudowany standard graficznego sprzęgu z użytkownikiem.

Na losy UNIX-a dodatkowy wpływ zaczyna wywierać lekceważony dotąd czynnik – standaryzacja. Na przykład POSIX 1003.1, przyjęty przez ANSI standard podstawowych funkcji systemowych, wyraźnie zyskał na znaczeniu odkąd władze federalne USA zdecydowały się nie kupować oprogramowania z nim niezgodnego. POSIX-owych grup roboczych jest w tej chwili już kilkanaście i standaryzują one niemal wszystko poza budową wewnętrzną systemu operacyjnego. Rodzi to obawy dwojakiego rodzaju: po pierwsze, proces standaryzacji może się nigdy nie skończyć, i po drugie, zbyt daleko idąca standaryzacja może skutecznie hamować innowacyjność.

Tę w sumie niezbyt klarowną sytuację dodatkowo komplikuje asekuranctwo większości firm, z reguły wstępujących zarówno do OSF, jak i do UI (dziś w skład OSF wchodzi sto kilkadziesiąt firm, korporacji, organizacji i uniwersytetów, w skład UI prawie sto). I nic dziwnego – obie strony mają swoje atuty: za UI przemawia gotowy SVR4 i UNIX-owe doświadczenie AT&T, za OSF – potęga takich gigantów, jak IBM czy DEC. W każdym razie o dalszych losach UNIX-a będą decydować liderzy obu tych ciał oraz – przede wszystkim – rynek.

JACEK SURMA

W następnych numerach: o SVR4, AIX-ie i Mach-u (2) oraz o POSIX-ie i innych standardach UNIX-a (3).

MIŁOSZ MUSZYŃSKI
Instytut Informatyki, Automatyki i Robotyki
Politechnika Poznańska

Fizyczne struktury systemów baz danych (2)

Techniki mieszania

Efektywne zastosowanie trie-drzew jest możliwe przy założeniu, że liczba wartości kluczy przechowywanych w bazie danych jest porównywalna z liczbą wszystkich możliwych wartości kluczy. Najczęściej jednak zbiór wszystkich możliwych wartości kluczy jest nieporównywalnie większy niż zbiór kluczy przechowywanych. Na tym spostrzeżeniu opierają się techniki mieszania.

Do podstawowych pojęć w tych technikach należą: **przestrzeń adresów** oraz **przestrzeń kluczy**. Przestrzeń adresów odpowiada zbiorowi wartości kluczy przechowywanych w bazie danych i jest z założenia wielokrotnie mniejsza niż przestrzeń kluczy, określana jako zbiór wszystkich możliwych wartości kluczy.

Podstawą technik mieszania jest działanie tak zwanej funkcji mieszającej, przekształcającej wartości kluczy w ich adresy. Jednemu adresowi – z przestrzeni adresów – odpowiada wiele wartości kluczy – z przestrzeni kluczy, ponieważ jednak prawdopodobieństwo przekształcenia dwu różnych wartości kluczy w ten sam adres jest małe, można więc wykorzystywać funkcję mieszającą do wyszukiwania, wstawiania i usuwania kluczy.

Przestrzeń adresów reprezentują elementy tablicy, w której są fizycznie umieszczane klucze. Tablica ta może być implementowana zarówno w pamięci operacyjnej, jak i w pamięci zewnętrznej, nie ma więc tu tak ścisłego podziału na struktury stosowane w pamięci operacyjnej i w pamięci zewnętrznej.

W tradycyjnych metodach mieszania zakłada się, że rozmiar tablicy, a więc wielkość przestrzeni adresów, jest stały. Są to techniki mieszania statycznego. Istnieją też techniki mieszania dynamicznego, w których jest możliwa modyfikacja rozmiaru tablicy. Te pierwsze są obecnie chętniej stosowane dla mniejszych zbiorów danych w pamięci operacyjnej, te drugie natomiast – przy zarządzaniu plikami dyskowymi. Podział taki nie wynika jednak z żadnych predyspozycji którejś z technik do danego rodzaju pamięci.

Techniki mieszania dynamicznego stanowią uogólnienie technik mieszania statycznego, z tego powodu podstawowe pojęcia są wspólne dla obu tych technik.

MIESZANIE STATYCZNE

Załóżmy, że rozmiar tablicy wynosi M, czyli że można w tej tablicy umieścić M kluczy. Przestrzeń adresów ma zatem postać przedziału liczb całkowitych $\langle 0, M\text{-}1\rangle$. Funkcja mieszająca $h(k)$ przekształca każdą z możliwych wartości klucza k w liczbę całkowitą z przedziału $\langle 0, M\text{-}1\rangle$. Wyszukiwanie klucza w tablicy odbywa się przez obliczenie wartości funkcji mieszającej. Przy takiej metodzie dostępu złożoność wyszukiwania jest rzędu $O(1)$, gdyż czas potrzebny do odnalezienia klucza nie zależy od liczby przechowywanych wartości kluczy.

Przy organizacji dostępu z wykorzystaniem funkcji mieszającej trzeba brać pod uwagę fakt, że nie dla wszystkich różnych par kluczy k_1 i k_2 zachodzi $h(k_1) \langle\rangle h(k_2)$. Sytuacja, w której $h(k_1) = h(k_2)$, nazywa się **kolizją**, biorące udział w kolizji klucze k_1 i k_2 nazywa się **synonimami**, a sposób postępowania po wystąpieniu kolizji nazywa się **algorytmem rozwiązywania kolizji**.

Algorytmy rozwiązywania kolizji umożliwiają zapamiętywanie i wyszukiwanie synonimów, jednak złożoność wyszukiwania przy częstym występowaniu kolizji staje się większa niż $O(1)$, w skrajnym przypadku może nawet stać się liniowa. Techniki mieszania zapewniają na ogół znacznie mniejszy średni czas wyszukiwania niż techniki oparte na strukturach drzewiastych, jednak czas wyszukiwania w najgorszym przypadku jest znacznie dłuższy.

Miarą efektywności stosowaną w technikach mieszania jest średnia liczba prób koniecznych do wyszukania klucza oznaczana przez C, przy czym przez próbę rozumie się dostęp do zapamiętanej wartości klucza i porównanie jej z argumentem wyszukiwania. Średnia liczba prób jest zazwyczaj funkcją współczynnika W wypełnienia tablicy, zdefiniowanego jako stosunek liczby przechowywanych wartości kluczy do liczby pozycji tablicy. Oprócz średniej liczby prób koniecznych do wyszukania klucza istotnym wskaźnikiem efektywności jest średnia liczba prób koniecznych do wyszukania wolnego miejsca w tablicy, interpretowana także jako średnia liczba prób przy wyszukiwaniu niepomyślnym i oznaczana przez C'. Ponieważ wyszukiwanie wolnego miejsca wymaga zawsze przejrzenia wszystkich synonimów danego klucza, więc $C' >= C$.

Na efektywność dostępu do danych w technikach mieszania zasadniczy wpływ mają dwa czynniki: dobór funkcji mieszającej oraz dobór algorytmu rozwiązywania kolizji.

Funkcje mieszające

Funkcja mieszająca powinna być możliwie efektywnie obliczana oraz powinna równomiernie rozmieszczać klucze w przestrzeni adresów. Pierwszy warunek wiąże się z koniecznością częstego obliczania funkcji mieszającej, najlepiej więc, jeżeli jej obliczenie sprowadza się do wykonania kilku podstawowych operacji na poziomie maszynowym. Drugi warunek wiąże się z wymaganiem, aby jednemu adresowi w przestrzeni adresów odpowiadała w przybliżeniu jednakowa liczba wartości kluczy z przestrzeni kluczy. Wymaganie to w praktyce jest jeszcze dodatkowo rozszerzane, ponieważ reprezentacje dwójkowe wartości kluczy występujących z największym prawdopodobieństwem w praktyce nie mają zazwyczaj rozkładu jednostajnego. Niejednostajność rozkładu wynika z częstego występowania pewnych grup kluczy:

- grupy kluczy tworzących ciągi arytmetyczne, np. *TAB1*, *TAB2*, *TAB3*,...
- grupy kluczy z tym samym znakiem na określonej pozycji,
- grupy kluczy z ograniczonym repertuarem znaków.

Częste występowanie powyższych grup kluczy eliminuje z użycia takie funkcje mieszające, których obliczenie polega na wycięciu z reprezentacji dwójkowej klucza odpowiedniej liczby bitów. Wszystkie bity reprezentacji dwójkowej adresu wyliczonego przez funkcję mieszającą powinny być uzależnione od wszystkich bitów reprezentacji klucza. Spełnienie tego warunku jest konieczne po to, aby funkcja mieszająca nie generowała grup tych samych adresów przy niejednostajnym rozkładzie kluczy w przestrzeni kluczy.

Do najbardziej znanych metod konstruowania funkcji mieszających należą metody [2, 4, 9]:

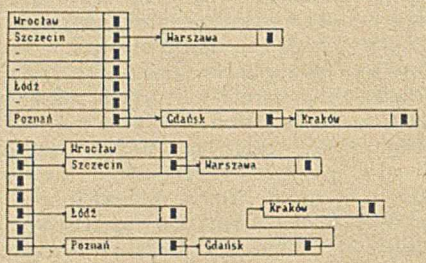
- dzielenia,
- mnożenia,
- środka kwadratu,
- składania,
- dzielenia wielomianów,
- przekształcania systemów liczbowych.

Są to metody niezależne od rozkładu kluczy w przestrzeni kluczy. Istnieją też metody tworzenia funkcji mieszających w zależności od przewidywanego lub danego a priori rozkładu kluczy; jest to mieszanie optymalne, z funkcją mieszającą, generującą minimalną liczbę kolizji [8].

Algorytmy rozwiązywania kolizji

Można wyróżnić dwie zasadnicze grupy metod rozwiązywania kolizji [2]:
– metody, w których synonimy zapamiętuje się w dodatkowych obszarach pamięci poza tablicą,
– metody rozwiązujące problem zapamiętywania synonimów w granicach tablicy.

Metody należące do grupy pierwszej są proste oraz umożliwiają przechowywanie większej liczby kluczy niż liczba pozycji tablicy. Wadą tych metod jest mniejsze wykorzystanie pamięci. Obszary nadmiarowe organizuje się najczęściej w postaci list, więc niezbędne jest rozszerzenie pozycji tablicy o pola wskaźników do list synonimów (rys. 1). Średnia liczba prób dla metod tej grupy jest mała i wynosi $C = 1 + W/2$, przy czym W może być większe niż 1 ze względu na istnienie dodatkowych obszarów nadmiarowych.



Rys. 1

Stosowanie metod rozwiązujących problem zapamiętywania synonimów w granicach tablicy, ogranicza się do sytuacji, w których liczba przechowywanych kluczy jest mniejsza lub równa liczbie pozycji tablicy, czyli gdy $W = 1$. Rozróżnia się dwie podstawowe metody tej grupy: metodę łańcuchów oraz metodę otwartego adresowania.

Metoda łańcuchów jest analogiczna do metody tworzenia list synonimów w obszarach nadmiarów, z tą różnicą, że w metodzie łańcuchów listy synonimów tworzy się w granicach zasadniczej tablicy (rys. 2). Ciągi wskaźników buduje się od adresu wyznaczonego przez funkcję mieszającą $h(k)$ do komórki, w której został umieszczony klucz k. W tym celu każda pozycja tablicy jest wyposażona w pole wskaźnika. Do poszukiwań wolnego miejsca w tablicy można stosować różne metody, m.in. omówione dalej metody otwartego adresowania. Metoda łańcuchów jest bardzo szybka i dodatkowo ma tę zaletę, że dopuszcza stuprocentowe wypełnienie tablicy. Przy takim całkowitym wypełnieniu ($W = 1$) współczynniki C i C' wynoszą odpowiednio 1,8 i 2,1.

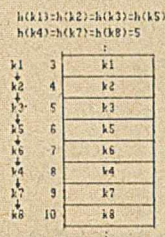
Rys. 2

Metoda otwartego adresowania polega na wyznaczeniu i wykorzystaniu ciągu funkcji mieszających $h_0, h_1, h_2, h_2, \ldots$ Funkcja h_0 jest zwykłą funkcją mieszającą, następne funkcje ciągu służą natomiast do obliczania kolejnych adresów w tablicy, jeśli na pozycji wyznaczonej w poprzednim kroku wystąpiła kolizja. Poszczególne odmiany otwartego adresowania różnią się sposobem tworzenia kolejnych funkcji mieszających ciągu. Do najważniejszych odmian metody otwartego adresowania należą: metoda liniowa, ważonego przyrostu, kwadratowa, podwójnego mieszania oraz ilorazowa.

Metoda liniowa jest jedną z najprostszych i najczęściej stosowanych odmian metody otwartego adresowania. Kolejne funkcje mieszające wyznacza się na podstawie wzoru:

$$h_i(k) = (h_0(k) + C \cdot i) \bmod M \quad c = \text{const}.$$

Stosowanie metody liniowej sprowadza się do przeszukiwania tablicy z pewnym stałym krokiem c. Warunkiem koniecznym sprawnego działania metody jest, aby liczba c nie była podzielnikiem liczby pozycji tablicy M. Jeżeli warunek ten nie jest spełniony, to wyszukiwanie wolnych miejsc w tablicy obejmuje zaledwie c/M obszaru tablicy. Najpoważniejszą wadą metody liniowej jest występowanie zjawiska grupowania pierwotnego. Polega ono na łączeniu się grup synonimów, co wydłuża średni czas wyszukiwania (rys. 3).



Rys. 3

Metoda ważonego przyrostu stanowi modyfikację metody liniowej. Zaletą tej metody jest brak zjawiska grupowania pierwotnego. Kolejne funkcje mieszające generuje się według wzoru:

$$h_i(k) = (h_0(k) + q(k) \cdot 1) \bmod M,$$

przy czym $q(k)$ zapewnia zmienny krok przeszukiwania w zależności od wartości klucza. Dla każdej wartości klucza k liczby $q(k)$ i M powinny być względnie pierwsze, w przeciwnym razie przeszukiwanie, podobnie jak w metodzie liniowej, nie będzie obejmowało całej tablicy.

Inną modyfikację metody liniowej dokonaną w celu wyeliminowania zjawiska grupowania pierwotnego stanowi **metoda kwadratowa**, określona wzorem:

$$h_i(k) = (h_0(k) + c \cdot i + d \cdot i^2) \bmod M \mid c = \text{const}, \; d = \text{const}.$$

Wadą tej metody jest występowanie zjawiska grupowania wtórnego. Połączenie grup synonimów następuje wówczas, gdy przecięcie ścieżek odpowiadających listom synonimów nastąpiło w tych samych krokach działania algorytmu (czyli dla takiej samej wartości i). Grupowanie wtórne zachodzi dla każdej metody otwartego adresowania, w której z równości

$$h_i(k_1) = h_i(k_2)$$

wynika równość

$$h_{i+1}(k_1) = h_{i+1}(k_2).$$

Metoda podwójnego mieszania dająca w praktyce najlepsze wyniki, polega na zastosowaniu dwu różnych funkcji mieszających, $h_0(k)$ i $g(k)$, w sposób przedstawiony za pomocą wzoru:

$$h_i(k) = (h_0(k) + g(k) \cdot i) \bmod M.$$

Jeżeli liczba pozycji tablicy M jest liczbą pierwszą, to poleca się następujące funkcje:

$$h_0(k) = k \bmod M, \quad g(k) = 1 + k \bmod (M-2).$$

Można też zastosować dowolne funkcje spełniające warunek

$$h_0(k) = M - g(k).$$

Zaletą **metody ilorazowej** jest prostota obliczeń przy dość dobrych właściwościach. Metodę tę można przedstawić za pomocą wzoru:

$$h_i(k) = (k + \text{entier}\,(k/M) \cdot i) \bmod M$$

Należy zaznaczyć, że wszystkie wartości generowane przez funkcje mieszające muszą znajdować się w przedziale $\langle 0, M-1 \rangle$. Dzielenie modulo M jest zatem integralną częścią każdego obliczenia funkcji mieszającej.

Usuwanie kluczy

W strukturach drzewiastych usuwanie było problemem równoważnym wstawianiu, w technikach mieszania usuwanie jest natomiast problemem o wiele bardziej złożonym niż wstawianie. Przyczyną tego

jest konieczność pamiętania zależności między synonimami. Każdy usuwany klucz może być potencjalnym ogiwem łączącym synonimy i jego brak może spowodować bezpowrotną utratę możliwości odszukania pewnych kluczy. Problem usuwania przedstawia się odmiennie dla różnych metod rozwiązywania kolizji. Można wyróżnić dwie zasadnicze grupy metod rozwiązywania kolizji ze względu na trudność operacji usuwania:

- metody z jednoznacznymi związkami między zajętymi pozycjami tablicy,
- metody z wieloznacznymi związkami między zajętymi pozycjami tablicy.

Rzeczywiste usunięcie elementów jest możliwe jedynie przy zastosowaniu metod grupy pierwszej, jednoznaczne związki między pozycjami umożliwiają bowiem modyfikację tablicy, polegającą na przemieszczeniu zawartości wszystkich pozycji tablicy występujących za pozycją usuniętą. W metodach grupy drugiej wieloznaczne związki pomiędzy pozycjami tablicy uniemożliwiają rzeczywiste usunięcie klucza. Jest możliwe jedynie usunięcie pozorne, polegające na wpisaniu w miejsce usuwanego klucza specjalnego kodu, który podczas przeszukiwania tablicy powoduje przejście do następnego elementu przeszukiwanej grupy synonimów, a przy wstawianiu nowego klucza oznacza pozycję pustą. W ten sposób każda pozycja tablicy może znajdować się w jednym z trzech stanów: wolnym, zajętym lub usuniętym. Ze stanu usuniętego jest możliwe tylko ponowne przejście do stanu zajętego. Rozwiązanie takie ma jednak tę wadę, że w razie zmniejszenia się liczby zajętych pozycji tablicy czas przeszukiwania pozostanie stały, gdyż w dalszym ciągu trzeba będzie przeszukiwać elementy znajdujące się w stanie usuniętym.

Do metod z wieloznacznymi związkami między zajętymi pozycjami tablicy zalicza się wszystkie metody otwartego adresowania, z wyjątkiem metody liniowej. Metoda liniowa, metoda łańcuchów oraz metody z dodatkowymi obszarami pamięci synonimów mają jednoznaczne związki między zajętymi pozycjami tablicy. We wszystkich metodach mieszania, bez względu na rodzaj związków między zajętymi pozycjami tablicy, usuwanie jest operacją wielokrotnie bardziej złożoną od operacji wstawiania i wyszukiwania.

Istotną wadą przedstawionych metod mieszania jest ich statyczność. Metody raz zaprojektowane na etapie implementacji systemu nie mogą być modyfikowane podczas jego użytkowania. Jeżeli na etapie implementacji przyjmie się wielkość tablicy w taki sposób, aby odpowiadała ona maksymalnemu przewidywanemu rozmiarowi zbioru danych, to efektywność wykorzystania pamięci przez większość czasu użytkowania będzie mała. Jeżeli natomiast przyjmie się wielkość tablicy zgodną z przewidywanym początkowym zapotrzebowaniem, to po pewnym czasie wystąpi potrzeba jej zwiększenia i konieczność kosztownej reorganizacji całej tablicy. Innym problemem jest charakterystyczna dla metod mieszania zależność efektywności dostępu od współczynnika wypełnienia tablicy. Konieczność zwiększenia tablicy pojawia się zazwyczaj nie z powodu jej fizycznego przepełnienia lecz z powodu nadmiernego wydłużenia czasu dostępu do danych. Zwiększenie tablicy pociąga za sobą konieczność zmiany funkcji mieszającej, a ponieważ zmieniona funkcja nie odpowiada dotychczasowemu rozmieszczeniu kluczy, konieczna jest całkowita reorganizacja tablicy, równoważna pod względem kosztów z budową całej tablicy od początku.

MIESZANIE DYNAMICZNE

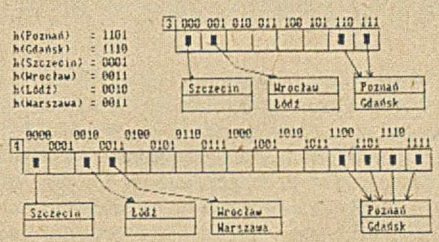
Techniki mieszania dynamicznego pozwalają na zmiany rozmiaru zbiorów danych o wiele rzędów wielkości, a przy tym zachowują charakterystyczny dla technik mieszania krótki czas wyszukiwania. Tradycyjne mieszanie statyczne wydaje się być techniką, która, poza przypadkami całkowicie statycznych zbiorów danych oraz niewielkich zbiorów danych przechowywanych w pamięci operacyjnej, stopniowo przestanie być stosowana. Od wczesnych lat siedemdziesiątych opracowano wiele odmian mieszania dynamicznego. Początkowe pomysły były związane z koncepcją mieszania rozszerzalnego (ang. *extendible hashing*), pod koniec lat siedemdziesiątych opracowano techniki mieszania liniowego (ang. *linear hashing*).

Techniki mieszania dynamicznego zostaną omówione w kontekście ich zastosowań jako struktur danych w pamięci zewnętrznej, dlatego zamiast o pozycjach w tablicy będzie mowa o plikach, blokach i komorach. **Bloki** są podstawową jednostką pamięci zewnętrznej, zorganizowanej jako zbiór plików, **komory** natomiast są łańcuchami bloków. Każdy adres wyznaczony przez dynamiczną funkcję mieszającą odpowiada jednej komorze i wchodzącym w skład tej komory blokom. Komory umożliwiają wypełnienie pamięci w stopniu większym od jedności, co zakłada się w większości metod mieszania dynamicznego. Bloki i komory w mieszaniu dynamicznym są odpowiednikami pozycji tablicy z zewnętrznymi listami synonimów w mieszaniu statycznym. Podstawową różnicę stanowi tutaj fakt, że jeden blok mieści wiele wartości kluczy, podczas gdy jedna pozycja tablicy mieści zazwyczaj tylko jedną wartość klucza. Różnica ta jest jednak mało istotna, bowiem równie dobrze można sobie wyobrazić metodę mieszania statycznego z blokami i komorami zamiast pozycji i list, jak i metodę mieszania dynamicznego z tablicą i listami synonimów zamiast bloków i komór.

Mieszanie rozszerzalne

Techniki mieszania rozszerzalnego należą do najstarszych technik mieszania dynamicznego [1]. Podstawowym zagadnieniem, związanym z technikami mieszania dynamicznego, jest sposób postępowania wobec zwiększającej się liczby rekordów w zarządzanym pliku. W mieszaniu rozszerzalnym dopuszcza się określony stopień wypełnienia komór pliku. Po przekroczeniu wartości granicznej wypełnienia komory tworzy się nową komorę, do której jest przemieszczana część rekordów z przepełnionej komory.

Dodatkowa struktura pośrednia, zwana katalogiem, umożliwia wyszukiwanie rekordów w komorach za pomocą funkcji mieszającej, mimo zmieniającej się liczby komór. Przykładowe struktury pliku zarządzanego techniką mieszania rozszerzalnego przedstawiono na rys. 4. Rozmiar katalogu jest zawsze równy 2^L. Liczba L jest wskaźnikiem rozdzielczości katalogu i jest równa liczbie bitów potrzebnych do zidentyfikowania jednej pozycji katalogu. Na rys. 4 widać dwie fazy rozszerzenia tego samego pliku, w którym katalogi mają wskaźniki rozdzielczości o wartościach odpowiednio $L=3$ i $L=4$.



Rys. 4

Każdy element katalogu zawiera wskaźnik do jednej z komór pliku; każda komora wskazywana może być przez jeden lub więcej wskaźników stanowiących elementy katalogu. W każdej komorze znajdują się rekordy, dla których funkcja mieszająca generuje wartości odpowiadające elementom katalogu wskazującym na tę komorę. Na przykład, w górnej części rys. 4 komora zawierająca rekordy *Poznań* i *Gdańsk* zawiera wszystkie rekordy pliku, dla których wartość funkcji mieszającej wynosi 110 lub 111. Ściślej rzecz biorąc, zawiera ona rekordy, dla których trzy najbardziej znaczące bity wartości funkcji mieszającej są równe 110 lub 111. Tyle bowiem bitów potrzeba do jednoznacznego zidentyfikowania elementu katalogu o poziomie rozdzielczości równym trzy. Przy próbie wstawienia do pliku rekordu z kluczem o wartości *Warszawa* okazuje się, że komora (001), w której powinien znaleźć się ten rekord, jest już wypełniona. Ulega ona rozdzieleniu na dwie komory, którym odpowiadają oddzielne elementy katalogu i odmienne wartości funkcji mieszającej. Wskaźnik rozdzielczości katalogu ulega zatem również zwiększeniu. W dolnej części rys. 4 przedstawiono stan pliku po wstawieniu rekordu *Warszawa* i zwiększeniu rozdzielczości katalogu. Wstawienie kolejnego rekordu, dla którego funkcja mieszająca wygenerowałaby adres 0011, spowodowałoby dalszą konieczność zwiększenia poziomu rozdzielczości katalogu.

Z tego przykładu można wyciągnąć dwa wnioski dotyczące wymagań wobec funkcji mieszającej w technice mieszania rozszerzalnego:

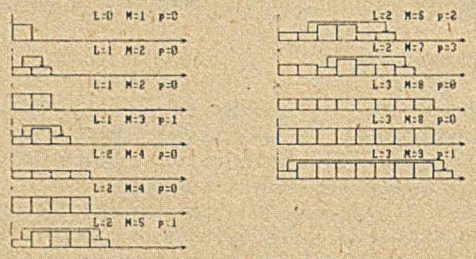
- funkcja mieszająca powinna generować adresy w postaci wielobitowych ciągów, z których w danej chwili wykorzystuje się L najbardziej znaczących pozycji,
- funkcja mieszająca musi być w dostatecznym stopniu losowa, w przeciwnym razie staje się przyczyną niepotrzebnego zwiększania rozmiaru katalogu.

Największą wadą mieszania rozszerzalnego jest konieczność istnienia katalogu. Zaletą tej metody jest dość duży stopień wykorzystania

pamięci oraz to, że operacje scalania i rozdzielania komór są wykonywane zawsze w odniesieniu jedynie do dwu komór jednocześnie, złożoność czasowa tych operacji jest więc mała.

Mieszanie liniowe

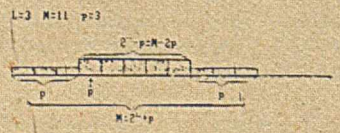
Podstawową zaletą mieszania liniowego w porównaniu z mieszaniem rozszerzalnym jest brak katalogu, ponadto rozdzielanie komór nie musi być związane z ich przepełnianiem.



Rys. 5

Podobnie jak w technikach mieszania rozszerzalnego, podstawą technik mieszania liniowego jest rozdzielanie komór, inne jest jednak kryterium wyboru komory rozdzielanej. Zamiast komór przepełnionych są rozdzielane kolejne komory pliku w porządku liniowym. Efektywność dostępu do pliku zarządzanego metodą mieszania liniowego zależy przede wszystkim od średniego współczynnika wypełnienia komór. Komory przepełnione, składające się z wielu bloków, są przyczyną drastycznego zmniejszenia efektywności dostępu. Wypełnienie komór jest optymalne wówczas, gdy każda komora składa się tylko z jednego bloku, a współczynnik wypełnienia pamięci jest równy jedności.

W metodzie mieszania liniowego rolę katalogu odgrywa wskaźnik oznaczony przez p. Wskaźnik ten zawiera zawsze numer komory, która ma być rozdzielona w najbliższym czasie. Na rysunku 5 zilustrowano proces rozszerzania pliku. W chwili zainicjowania plik składa się z jednej komory. Wskaźnik p zawiera numer komory zerowej, gdyż jest ona pierwszą komorą w pliku przeznaczoną do rozdzielenia. Po rozdzieleniu część rekordów z pierwszej komory jest przemieszczana do nowo utworzonej drugiej komory. Wybór rekordów do przemieszczenia następuje przez ponowne obliczenie funkcji mieszającej dla kluczy wszystkich rekordów komory rozdzielanej. W wyniku powstaje plik składający się z dwu komór. Wskaźnik p jest zerowany i ponownie wskazuje pierwszą komorę pliku. Podwojenie liczby komór pliku nazywa się **pełnym rozszerzeniem pliku**. Po pewnym czasie, gdy plik składa się z dwu komór, a wskaźnik p wskazuje pierwszą komorę pliku, komora ta jest rozdzielana, a wskaźnik p zwiększany o jeden. Następnie jest rozdzielana komora druga, wskaźnik p jest zerowany, a plik składa się już z czterech komór. Liczba komór pliku w chwilach, gdy wskaźnik p wskazuje pierwszą komorę pliku, jest zawsze równa 2^L. Liczba L oznacza liczbę dokonanych pełnych rozszerzeń pliku i jest nazywana **poziomem pliku**. Po dwóch pełnych rozszerzeniach ($L=2$) poziom pliku jest równy 2, a liczba jego komór wynosi $2^2=4$. Liczba komór w pliku, gdy wskaźnik p nie jest równy zeru, wynosi 2^L+p (rys. 6). Decyzje o dokonywaniu rozdzieleń kolejnych komór mogą być podejmowane na podstawie różnych kryteriów, na przykład, po upływie określonego czasu od ostatniego rozdzielenia lub po osiągnięciu przez współczynnik wypełnienia pliku zadanej wartości.



Rys. 6

Bez względu na bieżącą liczbę komór i wartości wskaźnika p, algorytm mieszania liniowego musi zapewnić, w każdym stanie pliku, możliwość wyszukania rekordu o dowolnej wartości klucza. Niech $h_n(k)$ oznacza funkcję mieszającą generującą ciągi n-bitowe. W każdej chwili plik składa się z p par komór już rozdzielonych oraz z 2^L-p komór jeszcze nie rozdzielonych (rys. 6). Generowane przez funkcję haszującą $h_L(k)$ L-bitowe ciągi wystarczają do określenia, w której z p par komór lub w której z 2^L-p komór jest umieszczony rekord o danej wartości klucza. Do wyboru jednej z dwu komór z pary komór już rozdzielonych jest jednak potrzebny dodatkowy bit ciągu generowanego przez funkcję mieszającą. Fakt ten uwzględniono w uogólnionym wzorze na funkcję mieszającą:

$$h(k) = \begin{cases} h_L(k) & \text{jeśli } h_L(k) >= p \text{ (dla komór nierozdzielonych)} \\ h_{L+1}(k) & \text{jeśli } h_L(k) < p \text{ (dla komór rozdzielonych)} \end{cases}$$

Przykładem funkcji mieszającej dla mieszania liniowego może być następująca funkcja:

$$h(k) = \begin{cases} k \bmod 2^L & \text{jeśli } k >= p \\ k \bmod 2^{L+1} & \text{jeśli } k < p \end{cases}$$

Funkcję mieszającą $h_{L+1}(k)$ nazywa się funkcję rozdzielającą, ponieważ to ona dokonuje faktycznego rozdzielenia rekordów w ramach par komór rozdzielonych.

Omówioną odmianę techniki mieszania liniowego nazywamy zwykłym mieszaniem liniowym [6]. Jego główną wadą jest brak możliwości utrzymania równomiernego i bliskiego jedności współczynnika wypełnienia. Komory nierozdzielone zawierają średnio dwukrotnie większą liczbę rekordów niż komory już rozdzielone (rys. 5 i 6). Średnia efektywność zwykłego mieszania liniowego jest satysfakcjonująca, jednak w wielu zastosowaniach liczba rekordów w pliku stabilizuje się na długie okresy i punkt stabilizacji może wystąpić w takim momencie, w którym rekordy są bardzo nierównomiernie rozmieszczone w komorach.

Istnieje wiele modyfikacji techniki zwykłego mieszania liniowego, m.in. metoda pamięci spiralnej. Aby ją omówić, trzeba wprowadzić inny sposób opisu procesu rozszerzania pliku (rys. 7). Łączy się on z odmienną numeracją i ustawieniem komór w pliku. Zakłada się bowiem, że w wyniku rozdzielenia pierwszej komory, dwie komory są umieszczane na końcu pliku. W ten sposób następna komora do rozdzielenia znajduje się zawsze na początku pliku, a zatem p wskazuje zawsze pierwszą komorę pliku. Podstawową zaletą takiej interpretacji jest to, że ustawienie komór pliku zawsze odpowiada kolejności ich rozdzielania.

W celu uniknięcia znamiennych dla techniki zwykłego mieszania liniowego wahań efektywności, istotne jest zapewnienie stałego rozkładu wypełnienia komór pliku. Rozkład taki powinien być stały bez względu na liczbę komór w pliku oraz bez względu na poziom pliku i wartość wskaźnika p. W tym celu funkcja mieszająca powinna generować taki rozkład rekordów w komorach, aby ostatnia komora pliku zawierała w przybliżeniu połowę liczby rekordów zawartych w pierwszej komorze pliku. W związku z tym postawiono wobec metod mieszania liniowego dwa postulaty:

- pożądane jest równomierne rozmieszczenie rekordów w komorach; komory rozdzielane w dalszej perspektywie powinny mieć mniejszy współczynnik wypełnienia niż komory rozdzielane w bliższej perspektywie;
- pożądane jest takie rozmieszczenie rekordów w komorach, w którym średnie zapełnienie komór nie ulegałoby zmianom wskutek dokonywanych rozdzieleń.

Funkcję mieszającą spełniającą powyższe wymagania zastosowano w metodzie pamięci spiralnej:

$$h(k) = \begin{cases} 2^L * 2^{d(k)} & \text{jeżeli } 2^L * 2^{d(k)} >= p \\ 2^{L+1} * 2^{d(k)} & \text{jeżeli } 2^L * 2^{d(k)} < p, \end{cases}$$

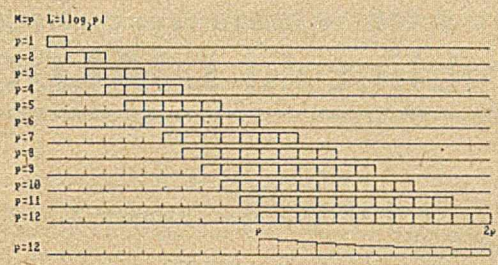
przy czym $d(k)$ jest funkcją przekształcającą w równomierny sposób zbiór kluczy w liczby z przedziału $<0,1)$. Efekt działania powyższej metody przedstawiono w dolnej części rys. 7.

Niech r oznacza wartość generowaną przez funkcję $h(k)$. Wzór określający współczynnik $f(r)$ wypełnienia r-tej komory ma postać:

$$f(r) = \frac{W}{\ln^2} \cdot \frac{p}{r}$$

przy czym W jest współczynnikiem wypełnienia pliku. Ze wzoru tego

wynika, że współczynnik wypełnienia komory jest odwrotnie proporcjonalny do numeru pozycji (r) tej komory w pliku. Wypełnienie pierwszej komory ($r = p$) jest więc dwukrotnie większe niż wypełnienie komory ostatniej ($r = 2p$). Tak więc przemieszczenie rekordów z pierwszej komory do dwu komór na końcu pliku nie spowoduje zmiany rozkładu wypełnienia komór pliku. Funkcja metody pamięci spiralnej ma zatem pożądane właściwości.

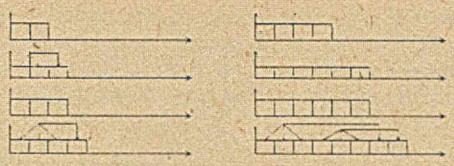


Rys. 7

Można obliczyć maksymalny współczynnik wypełnienia pliku, przy którym współczynniki wypełnienia wszystkich komór są mniejsze niż jeden. W pliku takim każda komora składa się tylko z jednego bloku. Ze względu na to, że pozycją o największym współczynniku wypełnienia jest pozycja pierwsza ($r = p$), można przyjąć, że $f(p) = 1$ i $W = f(p)*ln2 = ln2 = 0.6931$. Zatem przy średnim współczynniku wypełnienia $W = 0.6931$ wszystkie komory pliku mieszczą się w jednym pliku dyskowym.

Badania wykazały [3], że metoda pamięci spiralnej nie tylko usuwa niepożądane zmiany efektywności, lecz także generuje mniejszą ogólną liczbę nadmiarów niż średnio zwykłe mieszanie liniowe. Jest więc metodą atrakcyjną nie tylko ze względu na efektywność w najgorszym przypadku (co było przyczyną jej powstania), lecz także ze względu na efektywność średnią.

Zaletą rozwiązania, w którym rozdzielenie dotyczyło zawsze jednej komory pliku, jest prostota oraz efektywność samej operacji rozdzielania. Okazuje się jednak, że ogólna efektywność dostępu może zostać poprawiona przez zastosowanie takiej modyfikacji mieszania liniowego, w której rozdzielanie jest dokonywane w postaci serii rozszerzeń częściowych [5]. Przykładem może być metoda, w której każde pełne



Rys. 8

rozszerzenie pliku składa się z dwóch rozszerzeń częściowych. W pierwszym rozszerzeniu częściowym dwie komory są rozdzielane na trzy komory. Średnio po 1/3 rekordów z dwu rozdzielanych komór zostaje przemieszczonych do trzeciej komory. Po rozdzieleniu w ten sposób wszystkich komór pliku ogólna liczba komór ulega krotnemu zwiększeniu. Następnie jest wykonywane drugie częściowe rozszerzenie, podczas którego trzy komory są rozdzielane na cztery komory. Po zakończeniu drugiego częściowego rozszerzania liczba komór zwiększa się 4/3-krotnie, co po obu częściowych rozszerzeniach daje dwukrotny wzrost liczby komór. Dwa częściowe rozszerzenia składają się na jedno pełne rozszerzenie, po którym wskaźnik p jest zerowany i proces rozpoczyna się od początku (rys. 8).

Większość modyfikacji zwykłego mieszania liniowego sprowadza się do definiowania liczby komór rozdzielanych w jednym kroku i liczby komór utworzonych w wyniku rozdzielania [7]. Doprowadziło to z czasem do powstania koncepcji uogólnionego mieszania liniowego. W uogólnionym mieszaniu liniowym zakłada się, że liczba komór rozdzielanych jest równa s, a liczba komór powstałych w wyniku rozdzielenia jest równa t. Tak więc przy każdej operacji rozdzielania w miejsce s komór powstaje t komór. Zastępowanie s komór przez t komór jest interpretowane jako zastępowanie jednej komory liczbą komór równą t/s. Oznaczając t/s przez g można stwierdzić, że po każdym pełnym rozszerzeniu rozmiar pliku zwiększa się g-krotnie. Poziom pliku L można wówczas definiować jako $L = \log_g p$. Zwykłe mieszanie liniowe jest przypadkiem szczególnym uogólnionego mieszania liniowego dla $g = 2$.

Dla uogólnionego mieszania liniowego można zdefiniować także uogólnioną spiralną funkcję mieszającą:

$$h(k) = \begin{cases} g^L * g^{d(k)} & \text{jeżeli } g^L * g^{d(k)} >= p \\ g^{L+1} * g^{d(k)} & \text{jeżeli } g^L * g^{d(k)} < p \end{cases}$$

dokończenie na s. 10

ZPJ MIRANDA
ul. Jedwabnicza 1
62–700 Turek

PROPONUJE I INFORMUJE:

- Przyjmujemy oferty na kompleksową komputeryzację przedsiębiorstwa z uwzględnieniem:
  - zintegrowanego pakietu systemów (działających w sieci) do zarządzania przedsiębiorstwem; Aktualnie eksploatujemy system ODRA 1305 i rozproszone przetwarzanie na IBM PC.
- Zatrudnimy informatyków i projektantów systemów informatycznych.
- Sprzedamy na dogodnych warunkach:
  - czytnik kart CK 325-1 (1982 r.),
  - system monitorów ekranowych: JSG 7802 (1987 r.) z monitorami 7911 (1987 r.) – 6 szt.,
  - karty analityczne 80 kolumnowe – 1 000 000 szt.

Stan techniczny urządzeń jest bardzo dobry.

Wszelkie informacje można uzyskać w Dziale EPD Turek, tel. 55-20 wewn. 595, 597.

0/10/90

MARIUSZ POSTÓŁ
Zakład Badań Podstawowych
Elektrotechniki MP i PAN
Instytut Elektrotechniki
Warszawa

Przemysłowe systemy czasu rzeczywistego (2)

Struktura i funkcje biblioteki DIALOGUE

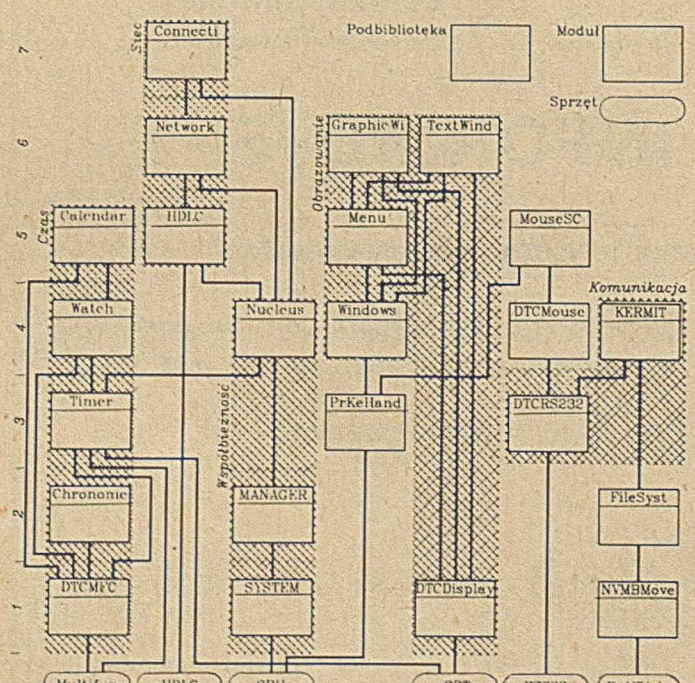
W drugiej części artykułu opisano strukturę i funkcje biblioteki DIALOGUE wraz z przykładem praktycznego jej zastosowania. Bibliotekę zaprojektowano na podstawie analizy zagadnień związanych z konstrukcją uniwersalnego, niezawodnego oprogramowania systemowego dla przemysłowych systemów czasu rzeczywistego. Wybrane wyniki analizy były przedmiotem dyskusji w pierwszej części artykułu.

Korzystając z wniosków zawartych w pierwszej części artykułu, opracowano zestaw modułów programowych – bibliotekę systemową, która upraszcza konstrukcję warstwy użytkowej programu i zwalnia od wielokrotnego tworzenia typowego oprogramowania systemowego.

STRUKTURA BIBLIOTEKI

Oferowane przez bibliotekę DIALOGUE moduły dostarczają dogodnych operacji umożliwiających:

- używanie kolorowej grafiki wysokiej rozdzielczości,
- komunikowanie się procesów realizowanych w różnych komputerach za pośrednictwem sieci lokalnej,
- komunikowanie się z użytkownikiem za pośrednictwem hierarchicznego menu,
- komunikowanie się z innymi komputerami za pośrednictwem kanałów transmisji szeregowej (RS232) i protokołu KERMIT,
- korzystanie z informacji o dacie i czasie astronomicznym,
- definiowanie operacji cyklicznych i czasu ich powtarzania,
- korzystanie z pamięci elektronicznych (dysk wirtualny z awaryjnym zasilaniem),



Struktura logiczna odwołań wzajemnych modułów biblioteki DIALOGUE, dla uproszczenia większość połączeń z modułem MANAGER została pominięta

- definiowanie oraz synchronizowanie wzajemne i z otoczeniem procesów współbieżnych.

Biblioteka DIALOGUE (rys.) ma budowę modularną, warstwową. Moduły, tworzące warstwę, korzystają tylko z obiektów zlokalizowanych w warstwach niższych. Najniższą warstwę stanowi sprzęt, widziany przez program jako zbiór zmiennych o adresach bezwzględnych, reprezentujących rejestry poszczególnych sterowników. Dodatkowo korzysta się tu z przerwań do synchronizowania przebiegu procesów sprzętowych z procesami programowymi.

Taka konstrukcja oprogramowania jest zgodna z zasadami programowania strukturalnego. Jej zastosowanie w praktyce napotyka jednak następującą trudność. Co zrobić w sytuacji, gdy moduł z warstwy wyższej musi zdefiniować operację, która ma być stosowana w warstwie niższej? Typowym przykładem takiej sytuacji jest definiowanie przez moduł użytkowy procedury, która ma być wykonywana synchronicznie z przerwaniem zegarowym obsługiwanym przez moduł biblioteczny. Przerwanie to jest również wykorzystywane przez moduły systemowe do wykonywania operacji związanych z czasem, takich jak uaktualnianie zegara systemowego i kalendarza, sterowanie realizacją procesów wstrzymanych na określony okres itd. Wymienione operacje powinny być realizowane przez oprogramowanie systemowe, ze względu na ich uniwersalny charakter. Do rozwiązania tego problemu Modula-2 udostępnia bardzo wygodny i efektywny mechanizm – **zmienne proceduro**we [2]. Korzystając z niego można zbudować bibliotekę, której moduły z warstw niższych nie importują bezpośrednio obiektów zdefiniowanych w modułach zlokalizowanych powyżej nich.

FUNKCJE BIBLIOTEKI

Biblioteka dostarcza operacji definiowania, inicjowania, synchronizowania i komunikowania się procesów współbieżnych wykonywanych w jednym procesorze.

Programowanie współbieżne

Moduły *SYSTEMSU* (SYSTEM SUpport) i *MANAGER* (rys.) zawierają zbiór operacji związanych z programowaniem współbieżnym. Moduł *SYSTEMSU* jest rozszerzeniem standardowego modułu *SYSTEM*, który zgodnie z definicją języka Modula-2 może zmieniać się w poszczególnych implementacjach.

Moduł *MANAGER* jest jedynym modułem, korzystającym z operacji programowania współbieżnego proponowanych przez język Modula-2. Dostępne w języku operacje są bardzo proste i umożliwiają jedynie bezpośrednie i natychmiastowe przełączanie procesora między procesami – również w wyniku odebrania przerwania. Takie mechanizmy semantycznie odpowiadają znanej z programowania sekwencyjnego instrukcji **goto** i nie spełniają wymagań programowania strukturalnego, a ich użycie wymaga wyjątkowych środków ostrożności. Z tego względu trzeba ograniczać obszar ich wykorzystania.

Do komunikacji procesów zastosowano koncepcję monitorów [7], zaimplementowaną w module *MANAGER*. Eksportuje on wszystkie niezbędne operacje do tworzenia, synchronizacji i kasowania procesów. Proponowane rozwiązanie jest podobne jak w języku Modula [13, 14], wprowadzono tu jednak wiele bardzo istotnych zmian. Zmodyfikowano sposób szeregowania procesów (wprowadzono priorytety procesów i podział czasu procesora), wbudowano mechanizm obsługi wyjątków

i zarządzania pamięcią tak, aby umożliwić zakończenie się procesu i w konsekwencji jego likwidację, rozróżniono dwa rodzaje synchronizacji z wywłaszczeniem i bez wywłaszczenia procesu wysyłającego sygnał, wbudowano mechanizm kontroli wykorzystania monitorów itd.

Kolejny moduł – *NUCLEUS* – przeznaczony również do wspomagania programowania współbieżnego, udostępnia mechanizmy komunikacji procesów z użyciem skrzynek pocztowych. Procesy korzystające z tej metody mogą tworzyć nową skrzynkę, wysyłać i odbierać wiadomości. Procesy czekające na przesyłkę mogą określać maksymalny czas oczekiwania, po którym zostaną wznowione, z sygnalizacją braku oczekiwanej przesyłki.

Sieć

Stosowanie rozproszonej struktury systemu wymaga udostępnienia mechanizmów komunikacji za pośrednictwem sieci lokalnej. W bibliotece DIALOGUE są one zrealizowane w modułach: *HDLC*, *Network* i *Connecti* (*Connections*). Konstrukcję tych modułów oparto na idei systemu otwartego ISO [9, 11]. Warstwę 1 (fizyczną) i warstwę 2 (łącza danych) zrealizowano z użyciem pakietu mikrokomputera komunikacyjnego zawierającego własny procesor, pamięć i programowalny sterownik transmisji szeregowej. W warstwie 2 użyto protokół *HDLC* [8]. Współpraca procesora głównego i procesora pakietu komunikacyjnego odbywa się za pośrednictwem równoległego kanału we-wy. Funkcję warstwy 3 (sieciowej) spełnia moduł *Network*, przy czym wprowadzono tu uproszczenie koncepcji adresowania i wykluczono połączenia wielokrotne. Przyjęto, że w rozważanej klasie zastosowań nie jest również potrzebny algorytm wybierania drogi przepływu pakietów.

Moduł *Connecti* zawiera funkcje warstwy 4 (transportowej) i 5 (sesji). Rolę tych warstw ograniczono tu do dostarczania dogodnych mechanizmów przesyłania dowolnych struktur danych bez potrzeby wnikania w techniczne lub ekonomiczne aspekty związane z samą transmisją. Jednym z zadań tego modułu jest testowanie wszystkich otwartych połączeń, w celu stwierdzenia w możliwie jak najkrótszym czasie braku operatywności lub błędów w pracy jednego z połączonych ze sobą partnerów-procesów.

Warstwa 6 (prezentacji) nie została zrealizowana, ponieważ nie przewiduje się potrzeby wystąpienia konwersji danych. Warstwę 7 (zastosowań) stanowią natomiast procesy użytkownika biblioteki.

Czas i data

Warstwa użytkowa programu, niezależnie od zastosowania, często wymaga istnienia operacji, pozwalających wyznaczać aktualną datę, czas astronomiczny lub wprowadzać opóźnienia w realizacji wybranych operacji. W systemach czasu rzeczywistego trzeba jednak liczyć się z pewnymi dodatkowymi wymaganiami. Typowo występuje tu konieczność wykonywania pewnych procedur w sposób cykliczny, ze zdefiniowanymi okresami ich powtarzania. Można dwojako rozwiązać to zagadnienie.

W pierwszym rozwiązaniu definiuje się kilka współbieżnych procesów i wykorzystuje dostępne operacje synchronizacji. Czas jest odmierzany przez systemowy proces *P1*, obsługujący przerwanie z zegara czasu rzeczywistego. Użytkownik natomiast definiuje procesy realizujące stosowne algorytmy, wymagające uwzględnienia relacji czasowych za pomocą operacji wstrzymującej proces na określony czas. Po upływie żądanego czasu wstrzymany proces jest uaktywniany przez sygnał, wysyłany z procesu *P1*. Wadą tego rozwiązania jest to, że od wysłania sygnału do faktycznego rozpoczęcia wymaganych operacji może upłynąć pewien okres, którego długość zależy od przyjętego algorytmu szeregowania i jest na ogół zmienna.

Drugie rozwiązanie polega na umożliwieniu użytkownikowi zdefiniowania procedury, która będzie wykonywana jako przedłużenie operacji obsługi przerwania. Rozwiązanie to nie ma poprzedniej wady, ale pozwala na praktycznie nieograniczone wydłużenie przez użytkownika czasu obsługi przerwania. W konsekwencji może to spowodować „niezauważanie" niektórych przerwań zegarowych. Ze względu na komplementarność oba mechanizmy są udostępnione warstwie użytkowej oprogramowania.

Dostęp do wymienionych informacji i operacji zapewniają moduły *Calendar*, *Clock*, *Timer*, *Chronome* (*Chronometer*) i DTCMFC (*DTC Multifunction Card Control*). Dodatkowo umożliwiają one wstrzymywanie procesów i uruchamianie ich w zadanym dniu i o zadanej godzinie oraz dostarczają dogodnych operacji konwersji danych.

Komunikacja z dyspozytorem

Ważnym zagadnieniem dla typowego systemu przemysłowego jest współpraca z dyspozytorem kontrolującym prawidłowość pracy obiektu. W nowoczesnych dyspozytorniach coraz częściej stosuje się ekrany monitorów do zobrazowania informacji i klawiatury (standardowe lub specjalistyczne) wraz z urządzeniami dodatkowymi, jak mysz lub pióro świetlne do wprowadzania danych. Stosuje się również monitory z naniesioną na ekran matrycą przełączników dotykowych [3]. Wymienione urządzenia zewnętrzne komputera pozwalają zaspokoić wszystkie wymagania stawiane w większości zastosowań.

W systemie DIALOGUE kilka modułów wspomaga komunikację z dyspozytorem. Przyjęto, że typowo będą używane dwa urządzenia do wprowadzania danych: klawiatura alfanumeryczna i mysz. Dane mogą być obrazowane na monitorach monochromatycznych lub kolorowych o różnej rozdzielczości.

Ze względu na to, że operator może korzystać jednocześnie z dwóch urządzeń do wprowadzenia danych, system musi zapewniać łączenie danych wprowadzanych tymi dwoma torami. Operacja ta jest realizowana przez moduł *PrKeHand* (Printer and Keyboard Handlers), który udostępnia procedurę wstawiającą do bufora klawiatury dowolny ciąg znaków. W module *MouseSC* (*Mouse Strings Controler*) użytkownik może przypisać poszczególnym zmianom stanu myszy odpowiadające im ciągi znaków. Moduł ten śledzi zachowanie się myszy i wstawia do bufora klawiatury stosowne ciągi znaków. Dzięki temu użytkownik widzi mysz i klawiaturę jako jedno urządzenie. Poza tym moduł *PrKeHand* odpowiada za współpracę z drukarką. Połączenie w jednym miejscu operacji drukowania i czytania z klawiatury wynika tylko z tego, że oba te urządzenia korzystają ze wspólnego przerwania w użytym komputerze.

Podobnie z jednego przerwania korzystają kanały transmisji szeregowej (RS232), z których jeden jest przeznaczony do współpracy z myszą. Odczytane z myszy ciągi znaków interpretuje się w module *DTCMouse*, udostępniającym operacje umożliwiające łatwe określanie i śledzenie zmian stanu myszy.

Bardzo wygodnym sposobem wydawania poleceń i odpowiadania na pytania zadane przez komputer jest menu. Koncepcję tę zrealizowano w module *Menu*. Udostępnia on procedurę funkcyjną wyświetlającą na ekranie menu o zawartości i strukturze zdefiniowanej przez użytkownika i zwracającą rezultat w postaci jednoznacznego identyfikatora wybranego rozkazu. Zasadniczo rozwiązanie to wywodzi się z koncepcji zaproponowanej w systemie operacyjnym komputera Lilith [10, 11], przy czym wprowadzono do niego kilka udogodnień.

Uzupełnieniem przedstawionego toru wprowadzania danych jest ich obrazowanie na ekranie monitora, realizowane przez moduły: *DTDisplay* (*DTC Display control*), *Windows*, *TextWindow*, *GraphixWin* (*Graphix Windows*).

W zależności od potrzeb, komputer może być wyposażony w monitory kolorowe lub monochromatyczne o różnych rozdzielczościach. Moduł *DTCDisplay* jest odpowiedzialny za kontrolę pracy sterownika monitora w taki sposób, aby użytkownik (ale także i moduły systemowe) mogły niezależnie od typu monitora realizować swoje funkcje.

W torze obrazowania danych wykorzystano standard zaproponowany przez Wirtha [6, 15], oparty na koncepcji okien. Pozwala ona dokonać podziału fizycznego ekranu na kilka niezależnych, logicznych okien i traktować monitor podobnie jak biurko ze stosem kartek. Poszczególne kartki – okna – mogą być wyciągane na wierzch stosu, są wtedy całkowicie widzialne, lub tylko częściowo wystawać spomiędzy innych.

Podstawową trudnością przy realizacji tej koncepcji jest konieczność odtwarzania zawartości okna, gdy jakiś jego fragment zostanie odsłonięty. Są możliwe dwa rozwiązania. Pierwsze polega na zapamiętywaniu przez moduł zarządzający oknami informacji niezbędnej do odtwarza-

nia obrazu. Drugie natomiast polega na określaniu przez użytkownika operacji odtwarzania zawartości dla każdego z aktywnych okien. Przechowywanie obrazu ma dwie niedogodności: wymaga zarezerwowania dużego (zwłaszcza dla okien z grafiką kolorową) obszaru pamięci na bufor oraz zmusza do wykonywania operacji na dwóch ekranach jednocześnie (fizycznym i buforze). Ponieważ sposób dostępu do elementów obrazu na ekranie fizycznym jest z reguły różny od sposobu dostępu do elementów bufora, więc operacje te muszą być realizowane w odmienny sposób, co dodatkowo pochłania czas procesora. Odtwarzanie z użyciem bufora ma jednak tę zaletę, że użytkownik nie musi sam projektować tej operacji. Kompromisem między wymienionymi rozwiązaniami jest zapamiętywanie treści obrazu w postaci wektorowej, a więc zapamiętywanie tylko parametrów dla poszczególnych elementów obrazu. To rozwiązanie ma sens, gdy obraz tworzą w większości złożone figury geometryczne, jak okręgi, prostokąty, linie itp., a nie punkty. Biblioteka DIALOGUE pozwala dla każdego okna osobno wybrać dowolną z wymienionych metod odtwarzania.

Wyróżniono też dwa rodzaje okien: tekstowe i graficzne. Zarządzanie oknami tekstowymi odbywa się za pośrednictwem operacji zgromadzonych w osobnym module (*TextWindow*). Uwzględniono tu jedną metodę odtwarzania – buforowanie. Informacja tekstowa nie wymaga buforów o dużych rozmiarach i dlatego wydaje się, że wprowadzenie tak uniwersalnego rozwiązania, jak dla okien graficznych, jest nieuzasadnione.

Bazą dla modułów *TextWindow* i *GraphicWin* jest moduł *Windows*, który odpowiada za zarządzanie miejscem na ekranie i hierarchią okien.

W systemie, w którym jest wiele procesów wykonywanych współbieżnie, istnieje problem dzielenia klawiatury i myszy przez wiele procesów. Rozwiązano go przyjmując, że w danej chwili tylko jedno okno jest aktywne i wszystkie wprowadzane znaki są kierowane do procesu korzystającego z tego okna. Jeśli w aktywnym oknie nie została zainicjowana operacja czytania, to wprowadzane znaki są używane do sterowania ruchem kursora i zmian hierarchii okien. Powyższe operacje mogą być również inicjowane z procesów użytkownika.

Komunikacja z komputerem oddalonym

Większość systemów przemysłowych wymaga przesyłania do innych komputerów wybranych informacji o zachowaniu się kontrolowanego obiektu. Jedną z najprostszych metod jest przesyłanie danych za pośrednictwem sprzęgu szeregowego (RS232) i ewentualnie linii telefonicznej z odpowiednimi modemami komunikacyjnymi. Umożliwia to korzystanie z linii komutowanych, a zatem przyłączanie różnych komputerów, często znacznie oddalonych od systemu przemysłowego.

Zastosowanie sprzęgu szeregowego RS232 wymaga użycia protokołu określającego zasady przesyłania i pozwalającego na sprawdzanie poprawności transmisji. W omawianej wersji biblioteki DIALOGUE zastosowano protokół *KERMIT* [4]. Jedną z jego licznych zalet jest to, że istnieją implementacje praktycznie dla wszystkich popularnych komputerów. Oznacza to, że komputery powszechnego użytku nie muszą mieć ani specjalizowanego oprogramowania, ani dodatkowego wyposażenia sprzętowego do przesyłania danych.

Moduł *KERMIT*, który faktycznie składa się z kilku modułów, jest odpowiedzialny za realizację protokołu. Po zainicjowaniu proces zdefiniowany w tym module wchodzi w stan zdalnego sterowania i czeka na zlecenia od komputera oddalonego, które są mu przesyłane za pośrednictwem kanału transmisji szeregowej. Podstawowym wymaganiem protokołu *KERMIT* jest przesyłanie tylko takich danych, które mają strukturę pliku. Plik musi mieć nazwę, podawaną jako podstawowy parametr każdego zlecenia transmisji danych.

Gromadzenie danych

W systemach przemysłowych należy unikać stosowania magnetycznych pamięci masowych. W zamian stosuje się pamięci elektroniczne zbudowane z układów CMOS. Stosowanie pamięci elektronicznej pozwala na natychmiastowy dostęp do dowolnego obszaru, nie wymaga podziału na sektory, ścieżki itd. oraz zapewnia znacznie większą niezawodność systemu. W zależności od konstrukcji pamięci elektronicznej może być ona dostępna przez procesor bezpośrednio, jako fragment pamięci operacyjnej, lub pośrednio przez operacje we-wy, jeśli jest sterowana przez autonomiczny układ logiczny. W bibliotece DIALOGUE moduły *FileSyst* (*File System*) i *NVMBMoves* są odpowiedzialne za korzystanie z systemu plików.

Ze względu na łatwość realizacji komunikacji między komputerami, najwygodniej jest operować strukturami typu plik. W programie natomiast stosuje się wiele różnych innych struktur danych. Rozwiązaniem eliminującym powyższą niedogodność jest umieszczenie w module *FileSyst* procedury:

PROCEDURE CreateFile (V: ADDRESS; Size: CARDINAL; FileName: ARRAY OF CHAR);

Definiuje ona zmienną o adresie *V* i rozmiarze *Size* jako plik o nazwie *FileName*. Dzięki temu moduł *KERMIT* może definicje te wykorzystywać przy przesyłaniu danych.

W omawianym rozwiązaniu dane są przechowywane w pamięci operacyjnej lub w pamięci dodatkowej, znajdującej się poza przestrzenią adresową procesora. Przesyłanie danych między pamięcią główną a dodatkową jest realizowane za pośrednictwem operacji z modułu *NVMBMoves*.

DALSZY ROZWÓJ BIBLIOTEKI

Biblioteka systemowa DIALOGUE została zrealizowana dla komputera składającego się z pakietów typu GSM, firmy GURU [5] i DSM, firmy Modulex [10], wykorzystujących standard magistrali BUSMAT II. W pierwszej wersji użyto pakietu procesora zbudowanego z użyciem układu firmy Zilog typu Z-80B. Obecnie biblioteka jest również dostępna dla pakietu z procesorem 16-bitowym firmy NEC typu V-20 (odpowiednik INTEL-8088).

Dalszy rozwój biblioteki idzie w kierunku implementacji protokołu *MAP* (*Manufacturing Access Protocol*) [12]. Protokół ten, jak należy sądzić, będzie odgrywał dominującą rolę w rozwoju oprogramowania i sprzętu sieciowego komputerów przemysłowych. Podstawową barierą, która uniemożliwiła pierwotnie implementację zgodnych z międzynarodowymi normami mechanizmów dostępu do sieci lokalnej, była dostępna pamięć operacyjna komputera głównego i koszty samej implementacji. W pierwszej wersji biblioteki dla komputera z procesorem Z-80, część odpowiedzialna za dostęp do sieci (moduły *Connecti*, *Network*, *HDLC* i *Nucleus*) wymagały 16 KB (czyli około 1/4 dostępnej) pamięci dla programu i 3 KB dla danych, nie licząc dalszych 16 KB dla programu realizowanego przez procesor pakietu komputera komunikacyjnego HDLC. Wymiana stosowanego procesora na V-20 oczywiście zniosła tę barierę i pozwoli na rozbudowanie oprogramowania sieciowego, choć i w obecnej postaci jest ono w pełni funkcjonalne i bardzo efektywne.

Planuje się również dalsze prace w kierunku rozwoju modułów realizujących protokół KERMIT. W pierwszej kolejności uzupełnienia będą dotyczyły operacji związanych z kartoteką i atrybutami plików, tak aby umożliwić ochronę przed odczytywaniem danych, znajdujących się w systemie, przez osoby niepowołane.

Ważnym kierunkiem rozwoju jest zwiększenie niezawodności przez zastosowanie sprzętowych mechanizmów ochrony zasobów.

PRZYKŁAD ZASTOSOWANIA

Bibliotekę systemową DIALOGUE wykorzystano do zaprojektowania i wdrożenia komputerowego systemu wspomagania pracy dyspozytora układu technologiczno-energetycznego Kopalni Węgla Brunatnego w Bełchatowie. Podstawowe zadania tego systemu to:

- informowanie o bieżącym zużyciu energii elektrycznej i poborze mocy przez całą Kopalnię i poszczególne urządzenia technologiczne,
- informowanie o efektywności pracy Kopalni i poszczególnych urządzeń, tzn. wyliczanie na bieżąco wskaźników energochłonności,
- informowanie o ewentualnych przekroczeniach dopuszczalnych mocy 15-minutowych i ostrzeganie o możliwości wystąpienia takich przekroczeń,
- informowanie o wydajności, z jaką pracują poszczególne urządzenia,
- zbieranie różnych danych związanych z efektywnością pracy i zużyciem energii elektrycznej,
- przesyłanie zebranych danych do komputera oddalonego.

System ten pracuje w Kopalni jako drugi system wspomagania. Pierwszy [1], zainstalowany w 1985 r., dostarcza informacji o pracy urządzeń wydobywczych Kopalni. Docelowo planuje się, że obydwa systemy zostaną zastąpione jednolitym systemem dostarczającym informacji związanych zarówno z procesem wydobywania, jak i zużyciem energii. Cel ten zostanie osiągnięty pod koniec 1991 r. Pierwszym etapem jest realizacja omawianego systemu dla układu technologiczno-energetycznego.

Omawiany system komputerowy składa się z dwóch komputerów: komputera obiektowego i obrazującego. Zadaniem komputera obiektowego jest wykonywanie pomiarów sygnałów analogowych, rejestracja zdarzeń na podstawie sygnałów dwustanowych i przechowywanie danych. Komputer obrazujący natomiast przetwarza dane otrzymane z komputera obiektowego i przedstawia je na ekranach dwóch monitorów graficznych. Zapewnia on również przesyłanie danych do komputera oddalonego, za pośrednictwem sprzęgu szeregowego (RS232) z użyciem protokołu *KERMIT*.

Obydwa komputery wyposażono w jeden terminal, który służy do wprowadzania informacji o topologii sieci zasilającej urządzenia technologiczne, współczynnikach wzmocnień tworów pomiarowych, dacie i czasie, potrzebie przejścia na czas zimowy (letni) itd. Możliwość modyfikowania tych informacji w systemie jest chroniona wielopoziomowym systemem haseł, przed możliwością manipulowania danymi przez osoby niepowołane. Dodatkowo każda zmiana jest odnotowywana w raportach.

Bazę systemową dla oprogramowania użytkowego stanowi biblioteka DIALOGUE w wersji dla procesora Z-80.

Obecnie trwają prace nad modyfikacją systemu i jego rozbudową. Podstawowe cele to zwiększenie niezawodności i rozbudowa funkcjonalna. Struktura docelowego systemu zasadniczo nie różni się od omówionej w części pierwszej artykułu. W systemie zostaną zdublowane zarówno komputery obiektowe, jak i obrazujące, w celu zwiększenia niezawodności. Zostanie również dołączony komputer zapewniający komunikację z dyspozytorem i realizujący funkcje doradcze. Należy podkreślić, że w systemie tym zostaną wykorzystane pakiety procesorów 16-bitowych, co jednak nie będzie wymagało ani wymiany komputerów, ani konieczności modyfikowania oprogramowania systemowego.

Autor chciałby serdecznie podziękować dr. inż. Danielowi Arendtowi za pomoc przy opracowaniu koncepcji biblioteki DIALOGUE i za bezpośredni udział w jej implementacji oraz dr. Antoniemu Zajączkowskiemu za wnikliwą analizę treści niniejszego artykułu. Słowa podziękowania należą się również mgr. inż. Longinowi Pobiedze za współudział w implementacji protokołu *KERMIT* w Moduli-2. Prace były prowadzone częściowo w ramach CPBR 5.7.

LITERATURA

[1] Arend D., Postół M.: Realtime multiprogramming system for mine control centre. Microprocessors and Microsystems, Vol. 14, No. 1, pp. 39–46, 1990
[2] Arendt D., Postół M., Zajączkowski A. M.: Modula-2 Wydawnictwa NOT, Warszawa 1989
[3] Corbrigan N. B., Starkey J. D.: A Concurrent General Purpose Operator Interface. IEEE Trans. on Software Engineering, Vol. SE-10, No. 6, 1984
[4] de Cruz F.: Kermit Protocol Manual. Columbia University Center for Computing Activities, New York 1984
[5] Guru: Informator techniczny. Guru, Warszawa 1989
[6] Gutknecht J.: System Programming in Modula-2: Mouse and Bitmap Display. Institut für Informatik, ETH Zürich, 1983
[7] Hoare C. A. R.: Monitors: An Operating System Structuring Concept. Communication of ACM, Vol. 17, pp. 549–557, 1974
[8] ISO DIS8802.2 Logical Link Control. ISO Draft International Standard 8802.2, 1984
[9] ISO IS7498 Information Processing Systems – Open Systems Interconnection – Basic Reference Model. International Standard 7498, 1974
[10] Modulex: Informator techniczny. Modulex, Warszawa 1989
[11] Tanenbaum A. T.: Sieci komputerowe. Wydawnictwa Naukowo-Techniczne, Warszawa 1988
[12] Weston R. M.: Industrial Computer Networks and the Role of MAP. Part 1, Part 2. Microprocessors and Microsystems, Vol. 10, No. 7, Vol. 11, No. 1, 1986
[13] Wirth N.: Design and Implementation of Modula. Software Practice and Experience, Vol. 7, pp. 67–84, 1977
[14] Wirth N.: Modula: A Language for Modular Multiprogramming. Software Practice and Experience, Vol. 7, pp. 3–35, 1977
[15] Wirth N.: Programming in Modula-2. Springer-Verlag, Berlin 1985.

Microcomp

ZAKŁAD SYSTEMÓW MIKROKOMPUTEROWYCH

MiCOMP

40–045 Katowice, ul. Astrów 7
telefon 513-086, telefaks 518-628,
teleks 315687

Systemy teletransmisji danych ICL/ODRA

Integracja systemów ICL i ODRA z mikrokomputerami standardu PC i systemami NOVELL, UNIX SCO

procesor sieci rozległej WAN
MiCOMP 8075
(emulacja ICL 7503)

most NOVELL – ICL/ODRA
NetCROSS-FXBM

program teletransmisji danych **MiCROS-FXBM** (PC, rozproszone bazy danych, wersja sieciowa – NOVELL), skaner MPXSCAN-8007,

adaptery, modemy, testery, emulatory terminali.

Instalacje w największych systemach na terenie całego kraju!

E0/1124/89

POLSKIE TOWARZYSTWO INFORMATYCZNE

Ocena stanu polskiej informatyki

Mimo wyraźnego ożywienia na rynku komputerów osobistych w okresie liberalizacji przepisów celnych, obecny stan informatyki w Polsce jest alarmujący. Sprzęt, sprzedawany w Polsce pod nazwą komputerów osobistych „profesjonalnych", w krajach rozwiniętych stanowi pod względem wartości zaledwie kilkanaście procent bieżących zakupów sprzętu informatycznego. Tymczasem w Polsce nagminnie próbuje się rozwiązać problemy „normalnej" informatyki sprzętem pomyślanym jako uzupełnienie podstawowego arsenału środków informatyki. Jednocześnie bardzo nieliczne zakupy komputerów większej mocy nie mogły zapewnić właściwego rozwoju zastosowań informatyki. Informatyczne zacofanie Polski względem krajów Europy Zachodniej pogłębia się dramatycznie; rzutuje to wybitnie negatywnie na rozwój gospodarczy naszego kraju.

Wobec bezwzględnej dominacji dość prymitywnych komputerów osobistych i zupełnego braku maszyn o współczesnych walorach użytkowych, nie dysponujemy w Polsce większymi doświadczeniami w stosowaniu nowoczesnych środków i metod informatyki.

Jakość sprzętu komputerowego stosowanego w Polsce wyraźnie ustępuje normom światowym. Jakość elementów, podzespołów i wyrobów krajowych i innych RWPG-owskich jest żenująca, a ekonomiczne zasady importu z drugiego obszaru płatniczego preferują zakupy w firmach nie zapewniających odpowiedniej jakości.

Fatalny stan techniczny łączności i polityka taryfowa resortu penalizująca tych użytkowników łączy, którzy korzystają z nich do transmisji cyfrowej, powodują, że przy ogólnym deficycie mocy obliczeniowej stanowczo zbyt duża jej część jest dostępna tylko lokalnie. Do obsługi stanowisk wprowadzania danych kupuje się komputery z ogromnym nadmiarem zdolności funkcjonalnych, niemożliwych do wyzyskania na tych stanowiskach, co oczywiście niepotrzebnie zwiększa koszt informatycznego uzbrojenia takich stanowisk.

W dziedzinie produkcji sprzętu informatycznego brak w Polsce jakiejkolwiek racjonalnej myśli przewodniej. Nie istnieje u nas ani przemysł oprogramowania, ani usług informatycznych. Nadal jeszcze powszechnie usankcjonowane piractwo programów, a także obowiązujące przepisy podatkowe i mechanizmy kształtowania cen, uniemożliwiają powstanie tych kluczowych działów informatyki.

W wielu zastosowaniach praktycznych nie przywiązuje się należytej uwagi do problemów metodyki projektowania i wdrażania systemów, wykorzystywania oprogramowania narzędziowego, standaryzacji itp. Istniejące i używane w Polsce standardy o zasięgu lokalnym nie są w dostatecznym stopniu upowszechniane. W dalszym ciągu nie udało się rozpocząć prac nad standaryzacją tych problemów w skali całego kraju. Wyposażenie narzędziowe **programistów w krajach rozwiniętych daje im znaczącą przewagę nad programistami krajowymi.**

Na niezłym poziomie są natomiast badania teoretyczne, prowadzone w kilku przodujących ośrodkach akademickich, co znajduje swój wyraz w uznaniu, jakim cieszą się poszczególni informatycy, a nawet szkoły naukowe. Niestety, ten niewątpliwy sukces badawczy polskiej informatyki nie jest w żaden sposób dyskontowany.

Brak dostępu do komputerów o większej mocy uniemożliwia prowadzenie ważnych badań naukowych w wielu dziedzinach, a także rozwój wielu ważnych zastosowań.

Bardzo często zakup komputera klasy PC jest bezsensowny; do celów – do jakich go kupiono – jest potrzebny większy komputer, a do celów, do jakich się go ostatecznie stosuje, jest za drogi. Nawet w tych dziedzinach, w których zastosowanie komputerów klasy PC jest obecnie najbliższe rozsądku i dość powszechne (księgowość małych firm, proste prace inżynierskie), brak norm i standardów skazuje te zastosowania na wieczną wzajemną izolację.

W ostatnich latach popełniono wiele błędów podczas wdrażania programu zastosowań informatyki w szkolnictwie. Wady wykonawstwa komputera ELWRO Junior i nieudolny serwis spowodowały trudności w jego eksploatacji. Do chwili obecnej brak jest niemal zupełnie oprogramowania dydaktycznego. Powinien powstać konsekwentny program produkcji i rozpowszechniania oprogramowania dydaktycznego, finansowany przez Ministerstwo Edukacji Narodowej. Ważnym problemem jest brak odpowiednio przeszkolonych nauczycieli, którzy potrafiliby wykorzystać komputer w codziennej pracy dydaktycznej. Wprowadzanie przedmiotu *„Elementy informatyki"* powinno następować tylko w szkołach, w których istnieją po temu odpowiednie warunki.

Rzeczą pilną jest dokształcenie jak największej liczby nauczycieli, którzy staliby się zaczynem wprowadzania zastosowań informatyki w nauczaniu poszczególnych przedmiotów szkolnych. Istnieją tu rozsądne programy, powstałe zresztą przy współudziale członków Towarzystwa.

Niezwykle pozytywna rola małych firm w stworzeniu rynku komputerów osobistych w Polsce jest oczywista. Niemniej oczywisty jest także fakt, iż skupiły one gros utalentowanych i przedsiębiorczych informatyków i organizatorów. Mniej pozytywnym, lecz bodajże nieuchronnym skutkiem działalności tych firm jest znaczne pogorszenie morale informatyków zatrudnionych w dużych firmach państwowych – wynikające nie tylko z gorszych warunków płacowych, lecz także z mniej atrakcyjnej pracy i ze znacznie słabszej dynamiki państwowego pracodawcy. Jest więc rzeczą pilną stworzenie warunków zachęcających większe i mocniejsze firmy do trwalszego związania ich losów z rozwojem zastosowań informatyki. Od strony ekonomicznej cel ten można osiągnąć zachęciwszy takie firmy do wydatnego zwiększenia inwestycji w infrastrukturę działalności informatycznej. Służyć temu powinna polityka podatkowa, uznanie zakupu warsztatowych komputerów i oprogramowania narzędziowego za zakupy inwestycyjne o krótkim okresie amortyzacji (zgodnie z tendencjami światowymi, uznającymi trzyletni okres życia użytecznego), a także śmiała polityka kredytowa. Powinno temu też służyć otwarte i pełne angażowanie firm do realizacji ważnych zadań informatycznych. Należy w tym celu stworzyć ekonomiczne zachęty (i prawne gwarancje) do zwiększania zakresu działalności firm; oprócz handlu komputerami i półlegalnej adaptacji pakietów użytkowych, firmy te powinny podejmować się w znacznie szerszym zakresie projektowania i kompletowania systemów informatycznych, a nawet całościowych dostaw „pod klucz".

Z jednej strony niejasna sytuacja ekonomiczna, głęboki kryzys przemysłu, atrofia usług i brak kapitału, charakteryzujące obecną sytuację Polski, z drugiej – nadmiernie rozbudowane struktury organizacyjne nadbudowy „naukowo-badawczej" i z trzeciej – bardzo silne interesy grupowe istniejących struktur przemysłu państwowego tworzą niezwykle skomplikowaną, pełną sprzeczności sytuację wyjściową. Nakłada się na nią narosła przez dziesięciolecia skorupa mitów, przesądów i haseł propagandowych, utrudniająca rzeczową dyskusję o przyszłości. Dotyczy to niestety także informatyki. Do szczególnie szkodliwych poglądów należą trzy następujące:

- *„Dysponujemy świetną kadrą fachowców"* (w sytuacji gdy żaden pracujący w Polsce informatyk nie ma dostępu do współczesnych środków informatyki, prenumerata światowych czasopism została drastycznie ograniczona, a wiedzę czerpie się z popularnych opracowań).
- Swoista wersja hasła *„Polak potrafi"*, głosząca triumfy poszczególnych grup badawczych i firm, którym udało się skonstruować prototypowe kopie produktów od dawna dostępnych na świecie w sprzedaży wysyłkowej (stale zapomina się, że w tej klasie wyrobów liczy się nie konstruowanie prototypów czy nawet małych serii, lecz skuteczne opanowanie technologii produkcji masowej, tudzież zorganizowanie marketingu i serwisu; wszystko inne – to problemy dla gimnazjalistów).
- *„Eksport urządzeń informatyki świadczy o dobrym poziomie (lub sukcesie) polskiej informatyki"* – ani eksport, ani nawet produkcja urządzeń, zwłaszcza oddzielnych elementów, np. drukarek, nie świadczy o zastosowaniach informatyki, a tak naprawdę liczą się tylko zastosowania.

Hasła i mity tworzą klimat sprzyjający snom o potędze, która nadejdzie, gdy tylko dokona się cud. Najczęściej cudem tym ma być wyzwolenie przesiębiorczości indywidualnej. Z reguły przytacza się przy tym przykłady sukcesów różnych małych firm kalifornijskich, zapominając, że powstawały one i działały w kontekście rozbudowanego i bardzo sprawnego przemysłu elektronicznego, oferującego od ręki wszystkie potrzebne elementy w dowolnej ilości i asortymencie. W Polsce, gdzie brakuje praktycznie wszystkich elementów, od kostek do złącz i kabli, każdy pomysł wymaga zagwarantowania dostaw praktycznie wszystkich części – a to kosztuje, zajmując nieporównywalnie więcej czasu niż kupowanie ich w sklepach.

Dlatego tak ważne jest przestawienie myślenia publicznego o informatyce na paradygmat usługowy: nie wytwarzać informatycznych cudeniek, lecz obsługiwać konkretne dziedziny. Sukces w informatyce powinien polegać nie na tym, co informatyk wymyśli czy skonstruuje, lecz na tym, o ile pomoże komuś innemu w jego pracy. Prawdziwe powodzenie informatyki mierzy się bowiem zyskiem, jaki przynosi ona jej użytkownikom. Konkretnie oznacza to, że:

● Należy doprowadzić do powstania przemysłu usług informatycznych, a potem ze wszech miar popierać jego rozwój; im bardziej kompleksowe będą te usługi, tym lepiej. Trzeba prawdopodobnie zacząć od stymulowania przemysłu oprogramowania, co wymaga gruntownej modernizacji wielu przepisów (od prawa autorskiego do polityki podatkowej i kredytowej), a także intensywnego propagowania nowoczesnych metod projektowania i realizacji systemów informatycznych (na tę propagandę trzeba zapewne wyłożyć środki z kasy państwowej).

● Przemysł środków informatyki należy w dużym stopniu opierać na sprawdzonych podzespołach i elementach kupowanych w dużym stopniu za granicą. Obecnie nie ma bowiem sensu podejmować w kraju konkurencyjnej produkcji tych elementów. Uzasadnione ekonomicznie jest za to kompletowanie systemów specjalizowanych oraz uzupełniających istniejące instalacje. Możliwe jest tutaj wykorzystanie takich naturalnych atutów, jak lepsza znajomość potrzeb użytkowników, tańsza siła robocza oraz istniejący potencjał.

● Potrzebny jest tani, niezawodny i prosty sprzęt informatyczny punktów gromadzenia danych i obsługi klientów oraz sieci lokalnych łączących takie punkty. Chodzi o wyposażenie okienek pocztowych, kas bankowych i sklepowych, punktów sprzedaży i rezerwacji biletów itp. Najprostsze nawet komputery klasy PC są do tych celów zbyt rozbudowane, zwykłe terminale ekranowe zaś – za ubogie. Urządzenia takie, z rozmaitymi wariantami standardowego oprogramowania, mogą liczyć na duży zbyt – oczywiście pod warunkiem sprowadzenia ceny do skali, w której poczta czy PKO będą mogły je kupować w setkach tysięcy sztuk.

● Należy wyposażyć czołowe uczelniane instytuty informatyki w nowoczesne środki informatyki. Z całą świadomością opowiadamy się tu za elitarnością: bardziej w Polsce brakuje kilkudziesięciu młodych specjalistów wykształconych na najwyższym poziomie niż kilku tysięcy niedouczonych. Zwiększenie przeciętnych kwalifikacji kadr zawsze przychodzi szybciej, gdy wysiłek koncentruje się na najlepszych.

● Należy rozwiązać problem dostępu polskiej nauki do nowoczesnych komputerów o dużej mocy, w tym – superkomputerów. Warto przy tym pamiętać, że chodzi tu nie o udostępnienie Crayów każdej szkole inżynierskiej, lecz o dostęp do niezbędnej mocy obliczeniowej dla czołówki polskiej nauki. Mogą więc wchodzić w grę rozwiązania jednostkowe, niestandardowe, nie do powielenia itd. Należy dążyć do wydatnego osłabienia, a nawet zniesienia embargo na dostęp do tego rodzaju maszyn.

● Należy szybko przystąpić do opracowania norm i standardów informatyki użytkowej. Nie można dłużej tolerować chaosu w sprzętowych realizacjach polskiego alfabetu, dowolności form komputerowej księgowości itp. Podobnie pilna jest sprawa podstaw prawnych działalności informatycznej. Ochrona praw jednostki będącej przedmiotem systemów informatycznych, poufność transmisji danych, autorskie i rzeczowe prawa twórców oprogramowania systemów, sądowa uznawalność dokumentów komputerowych, odpowiedzialność cywilna i karna dostawcy systemu za błędy i szkody powstałe u użytkowników, procedury weryfikacyjne, instytucjonalizacja uprawnień zawodowych informatyków (na wzór uprawnień budowlanych czy lekarskich) – oto najbardziej oczywiste przykłady problemów wymagających pilnego załatwienia.

Maj 1900

Nowe książki

WYDAWNICTWA NAUKOWO-TECHNICZNE

Janina i Michał Jankowscy: Przegląd metod i algorytmów numerycznych – część 1. WNT, Warszawa 1988, wyd. II, nakład 4 tys. egz., s. 212, ISBN 83-204-1040-1 cz.1

Książka stanowi pierwszą część monografii poświęconej przeglądowi metod numerycznych. Omówiono w niej własności arytmetyki zmiennopozycyjnej oraz podstawowe metody i alegorytmy rozwiązywania zadań interpolacyjnych i aproksymacyjnych, przybliżonego obliczania całek oraz rozwiązywania równań różniczkowych zwyczajnych. Specjalną uwagę poświęcono ocenie numerycznej stabilności algorytmów i porównaniu ich efektywności ze złożonością obliczeniową odpowiednich zadań.
Książka jest przeznaczona dla programistów, pracowników nauki i studentów kierunków informatycznych.

Jan Bielecki: Turbo Pascal z grafiką dla IBM PC. WNT, Warszawa 1989, wyd. I, nakład 20 tys. egz., s. 324, ISBN 83-204-0944-6
W książce przedstawiono język programowania mikrokomputerów – Turbo Pascal w jego wersji dla mikrokomputera IBM PC. Dokładnie omówiono wszelkie konstrukcje języka i zestawy podprogramów standardowych, opisy słowne uzupełniając licznymi i szczegółowo skomentowanymi przykładami programów. Wiele miejsca poświęcono problematyce programowania operacji wejścia-wyjścia oraz zastosowaniom języka jako narzędzia do programowania zagadnień z zakresu grafiki komputerowej na mikrokomputerach rodziny IBM PC. Książka jest przeznaczona dla szerokiego kręgu Czytelników – użytkowników popularnych mikrokomputerów.

Brian W. Kernighan, P. J. Plauger: Narzędzia programistyczne w Pascalu. WNT, Warszawa 1989, wyd. I, nakład 10 tys. egz., s. 412, ISBN 83-204-1026-6
W książce opisano proces konstrukcji wielu programów pascalowych, stanowiących codzienne, praktyczne narzędzia wspomagające pracę programisty w zaawansowanym środowisku systemowym (autorzy prezentują programy usługowe zaczerpnięte z systemu Unix). Proces ten ilustruje stosowanie zasad programowania strukturalnego i zstępującego na wszystkich etapach budowy programu, a więc w trakcie jego projektowania, budowania i testowania. Omówiono także zagadnienia efektywności i przenośności programów. Autorzy książki są znanymi programistami – praktykami.
Książka jest przeznaczona dla każdego, kto chce stosować profesjonalne metody przy pisaniu programów komputerowych.

L. Banachowski, A. Kreczmar: Elementy analizy algorytmów. WNT, Warszawa, 1989 wyd. I, nakład 2 tys. egz., s. 192, ISBN 83-20-1147-5
Książka jest poświęcona matematycznej analizie algorytmów. Omówiono w niej dwa aspekty tej analizy: semantyczną poprawność i złożoność obliczeniową, przy czym podano podstawowe pojęcia, takie jak algorytm, wartościowanie, obliczanie itp. oraz szczegółowe przykłady analizy algorytmów. Rozważono trzy powszechnie znane zadania: sortowanie, selekcję i znajdowanie klas równoważności.
Książka jest przeznaczona dla programistów, projektantów systemów przetwarzania informacji, pracowników informacji, pracowników nauki zajmujących się informatyką oraz dla studentów kierunków informatycznych i matematycznych.

Jarosław Deminet: System operacyjny MS-DOS. WNT, Warszawa 1990, wyd. I, nakład 15 tys. egz., s. 176, ISBN 83-204-1257-9
Książka zawiera obszerny zestaw informacji o systemie operacyjnym MD-DOS, a także najważniejsze wiadomości o architekturze mikrokomputerów IBM PC/XT oraz PC/AT, dla których jest on przeznaczony. Omówiono wszystkie istotne elementy systemu – od podstawowych cech sprzętu aż do złożonych, niskopoziomowych operacji oraz programów obsługi urządzeń. Podano wiele informacji objaśniających zasady działania systemu, sposób realizacji poszczególnych poleceń i programów wchodzących w jego skład.
Książka jest przeznaczona dla szerokiego kręgu użytkowników mikrokomputerów osobistych.

Najpierw były jakieś kawałki oprogramowania zrobione przez dwóch przyjaciół dla własnych potrzeb na nieciekawym minikomputerze PDP/7. Potem niezbyt pewne narzędzie różnych hobbystów, przeważnie z kręgów akademickich. Dziś system operacyjny najróżniejszych komputerów – od mikro- po superkomputery – przedmiot konkurencyjnej walki wielkich firm przemysłu informatycznego, pierwszy i jak dotąd chyba jedyny, który wyszedł daleko poza firmę macierzystą i dla którego powstał problem opracowania normy niezależnej od architektur komputerowych.

Tak w skrócie przedstawia się historia UNIX-a; dodać do tego należy dziesiątki znaczących konferencji, sympozjów i wystaw rocznie, nie mówiąc o setkach pomniejszych, liczne książki i czasopisma, międzynarodowe i lokalne organizacje użytkowników od Stanów Zjednoczonych po Kuwejt.

Polscy użytkownicy i sympatycy UNIX-a, na razie jeszcze nieliczni, zaczynają również poszukiwać wzajemnych kontaktów, czego dowodem jest niniejszy dział UNIX-owy, na który uzyskaliśmy miejsce dzięki uprzejmości Redakcji INFORMATYKI. Pomysł prowadzenia takiego działu powstał na inauguracyjnym spotkaniu Sekcji UNIX-owej Polskiego Towarzystwa Informatycznego. O samej Sekcji piszemy nieco więcej w rubryce „Wiadomości".

Zamierzamy zamieszczać w naszym dziale to, co może zainteresować użytkownika UNIX-a: artykuły przeglądowe, artykuły o nowych rozwiązaniach technicznych, wiadomości o imprezach krajowych i zagranicznych, pomysły i porady techniczne. Palącym jest problem włączania naszych polskich instalacji UNIX-owych do międzynarodowych sieci. Na pewno ważny będzie problem „internacjonalizacji" UNIX-a w odniesieniu do naszych polskich warunków.

Ostatecznie jednak o zawartości działu będą decydować nasi Czytelnicy. Wszystkich „UNIX-owców" zapraszamy więc do współpracy: czekamy na materiały i pomysły.

/książki

The Design and Implementation of the 4.3BSD UNIX Operating System

Od kilkunastu lat w Uniwersytecie Kalifornijskim w Berkeley powstają kolejne edycje oryginalnego wariantu UNIX-a, znanego powszechnie jako Berkeley UNIX. Systemy z Berkeley, popularne zwłaszcza w wersjach VAX-owych, zyskały sobie u fachowców uznanie bodajże większe niż System V i od początku wpływały stymulująco na rozwój pozostałych wariantów UNIX-a. Monografia Lefflera, McKusicka, Karelsa i Quartermana opisuje wnętrze UNIX-a w edycji 4.3BSD (*Berkeley Software Distribution*), a więc najnowszej z tego źródła. Z racji swej tematyki nie jest to książka dla UNIX-owych nowicjuszy, przeznaczona jest głównie dla doświadczonych programistów, studentów wyższych lat informatyki i naukowców.

Książka składa się z pięciu części. Pierwsza, wprowadzająca, przypomina ewolucję UNIX-a, w tym zwłaszcza wariantu z Berleley z edycją 4.3BSD *Tahoe* włącznie, oraz szkicuje budowę, mechanizmy i funkcje jądra systemu. Kolejne trzy części są właściwie szczegółowym rozwinięciem tego szkicu: druga opisuje zarządzanie procesami i pamięcią oraz związane z tym struktury, algorytmy i funkcje jądra, trzecia dotyczy szeroko pojmowanego wejścia-wyjścia, a więc organizacji systemu plików i zasad współpracy jądra z urządzeniami zewnętrznymi, czwarta przedstawia gniazdka (*sockets*) – oryginalny mechanizm synchronizacji procesów – i pokazuje jak zrealizowano dominujący w UNIX-owym świecie protokół sieciowy TCP/IP. Ostatnia, piąta część, słabiej związana z poprzednimi, dość dokładnie opisuje akcje początkujące i kończące prace systemu. Wszystkie rozdziały kończą się ćwiczeniami i bibliografią. Całość uzupełnia słownik i indeks.

Mimo akcentów podręcznikowych książka jest przede wszystkim wyczerpującą i autorytatywną monografią jednego z najciekawszych wariantów UNIX-a. Równie dokładnego opisu mechanizmu gniazdek czy modyfikacji w systemie plików (*fast filesystem*) w ogóle dotąd nie było. Miejscami – zwłaszcza w części czwartej – poziom szczegółowości jest nawet zbyt wysoki, co chwilami utrudnia śledzenie istoty zagadnienia. Ta szczegółowość ma jednak swoje zalety: pozwala obserwować z jakim trudem pierwotnie proste i zwarte jądro UNIX-a wchłania kolejne rozszerzania. Dotyczy to przede wszystkim mechanizmów sieciowych, nie bez powodu sprawiających wrażenie sztucznie doczepionych do jądra.

Za przykład implementacji systemu służy wyłącznie wersja oryginalna, czyli VAX-owa (VAX-owych dyletantów nie powinno to zniechęcać, bo wcześniejsza wiedza o architekturze VAX-ów nie jest niezbędna). Pewien mankament stanowi jednak brak odniesień do innych implementacji, co widać szczególnie w części drugiej – nie wszystkich interesują chociażby idiosynkrazje VAX-owego zarządzania pamięcią. Nie wpływa to istotnie na wartość książki – jest to solidna, szczegółowa monografia jednego z ciekawszych systemów operacyjnych, bez wątpienia jedna z najcenniejszych pozycji w całej dotychczasowej literaturze UNIX-owej. Mimo coraz szybszego ostatnio rozwoju UNIX-a, książka ta raczej nieprędko straci swą użyteczność.

JACEK SURMA

UNIX jest zastrzeżonym znakiem handlowym AT&T.

The Design and Implementation of the 4.3BSD UNIX Operating System – Samuel J. Leffler, Marshall Kirk McKusick, Michael J. Karels, John S. Quarterman, Addison-Wesley Publishing Company, 1989, s. 471, ISBN 0-201-06196-1

Systemy otwarte i ich wrogowie, czyli co dalej z UNIX-em (1)

Rozłam w UNIX-ie

Jeszcze cztery lata temu w UNIX-owym świecie panował spokój i porządek. Główny nurt rozwoju wyznaczały powstające w laboratoriach AT&T kolejne edycje Systemu V, w środowiskach akademickich dominował wariant z Berkeley, oryginalny i inspirujący, lecz głównie VAX-owy, a najpopularniejszą mikrokomputerową odmianą UNIX-a stopniowo stawał się XENIX. UNIX-owa rodzina systemów operacyjnych miała za sobą kilkanaście lat rozwoju i mocno niepewną przyszłość. Pozornie nic nie zapowiadało zmian.

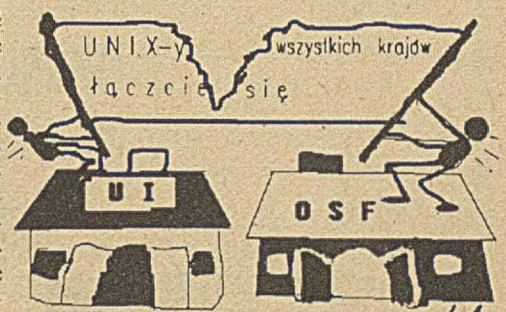
Być może kluczem do tego, co stało się potem, mogłoby być właśnie tytułowe pojęcie systemu otwartego. Ten ostatni przymiotnik ma – bez wątpienia również dzięki Popperowi – raczej pozytywne zabarwienie i pewnie dlatego często bywa nadużywany. Dotyczy to również takich informatycznych pojęć-metafor, jak *oprogramowanie otwarte*, *system otwarty* czy *otwarty system operacyjny*. A jednak niosą one pewną istotną treść, na tyle ważną dla zrozumienia tego, co dzieje się ostatnio w UNIX-owym świecie, by warto było spróbować ją wydobyć. **Systemem otwartym** byłby więc system komputerowy pozostający w zgodzie z uznanymi międzynarodowymi standardami, współpracujący z każdym produktem spełniającym wymagania tych standardów. W takim systemie pochodzące z różnych źródeł elementy – sprzęt, systemy operacyjne i programy użytkowe – mogłyby ze sobą dowolnie współdziałać, na przykład w sieci komputerowej, właśnie dzięki standaryzacji wszelkich połączeń między nimi. Systemy otwarte to na razie raczej odległa idea – nie jest zresztą pewne, czy możliwa do zrealizowania.

Natomiast już dużo łatwiej sobie wyobrazić **otwarty system operacyjny**. Musiałby to być system prawdziwie przenośny i uniwersalny, bez bliższych związków z jakąkolwiek wyróżnioną architekturą czy konfiguracją, o precyzyjnie zdefiniowanych i znormalizowanych regułach współpracy z oprogramowaniem użytkowym (tak w postaci źródłowej, jak i binarnej) oraz z użytkownikami (operatorem, administratorem, itp.). Proces rozwoju takiego systemu również powinien być otwarty, a więc jawny i oparty na współpracy wszystkich zainteresowanych. Nie ulega wątpliwości, że ze wszystkich dzisiejszych systemów operacyjnych UNIX jest tu stosunkowo najbliżej ideału.

Oczywiście w pomyśle otwartego systemu operacyjnego nowa jest prawdopodobnie tylko nazwa – użytkowników i programistów zawsze pociągały podobne idee. Nie miało to jednak do niedawna żadnego wpływu na strategię rynkową producentów komputerów: dominowały systemy jak najbardziej zamknięte, wyposażone w równie zamknięte, firmowe (ang. *proprietary*) systemy operacyjne. Na szczęście dla UNIX-a coraz powszechniejsza konieczność łączenia komputerów – czy to w sieci, czy w systemy rozproszone – sprawiła, że w firmach produkujących komputery poważnie zainteresowano się koncepcją systemów otwartych. Notowania UNIX-a zaczęły rosnąć.

W październiku 1987 roku opublikowano wiadomość, że nowa, zunifikowana wersja UNIX-a – *System V Release 4* (SVR4) – powstanie w ścisłej współpracy AT&T z firmą Sun Microsystems, znanym producentem komputerów. Konferencja prasowa na ten temat odbyła się dokładnie w dniu krachu na nowojorskiej giełdzie i zapewne dlatego nie wywołała większego zainteresowania. Ale kiedy w styczniu następnego roku przedstawiciele AT&T poinformowali o zamiarze zakupu pokaźnego pakietu akcji firmy Sun, natychmiast zorientowano się, czym to grozi. Na rynku systemów UNIX-owych duet AT&T-Sun znalazłby się w niezwykle uprzywilejowanej pozycji, zaś UNIX mógłby z czasem stać się zwykłym systemem firmowym, równie ściśle związanym z komputerami Sun, jak na przykład VMS z rodziną VAX-ów. Tym razem reakcja konkurencji była szybka. Grupa zaniepokojonych gigantów, m.in. IBM, DEC, Bull i Hewlett-Packard, powołała *Open Software Foundation* (OSF) – konsorcjum, które postawiło sobie za cel opracowywanie i promocję przemysłowych standardów oprogramowania, w tym w pierwszym rzędzie stworzenie nowego, wzorowanego na UNIX-ie, otwartego systemu operacyjnego.

Wraz z przystępowaniem do OSF kolejnych firm zaczęły pojawiać się pogłoski o ewentualnym akcesie AT&T. Kością niezgody miała okazać się decyzja OSF o przyjęciu AIX-a, raczej mało znanego, choć interesującego IMB-owskiego wariantu UNIX-a, za pierwowzór dla nowego systemu. Problem był podobno w tym, że AIX wywodził się z edycji 2 Systemu V, podczas gdy w AT&T uważano, że jakakolwiek nowa wersja UNIX-a powinna opierać się na najnowszej wówczas edycji 3.2. W każdym razie do porozumienia nie doszło. Z kolei zaczęto wyrażać wątpliwości co do czystości intencji OSF. Rzecz w tym, że wśród jej członków nie brakowało firm, które w przeszłości twardo broniły się przed częściowym choćby otwarciem swych własnych, zdecydowanie zamkniętych systemów. Niektórzy obawiali się więc, że OSF skutecznie zahamuje dalszy rozwój UNIX-a. Inni twierdzili, że jakikolwiek UNIX nie pochodzący z AT&T może być co najwyżej wariantem.



Ryc. LUDMIŁA ANDRUSIEWICZ

Wszystko to doprowadziło jeszcze w 1988 roku do zawiązania się stowarzyszenia firm komputerowych wiążących swe nadzieje z przyszłymi wersjami Systemu V i zamierzających wspierać AT&T w dalszych pracach nad tym systemem. Wśród jego założycieli znalazły się m.in. firmy takie, jak Olivetti, Unisys, Toshiba i Amdahl. Przyłączenie się duetu AT&T-Sun ugruntowało pozycję stowarzyszenia, które niebawem przekształciło się w niedochodową korporację *UNIX International* (UI). Z kolei w AT&T utworzono odrębną jednostkę organizacyjną, *UNIX Software Operation* (USO), która przyjęła całokształt spraw związanych z produkcją, rozwojem, sprzedażą i licencjonowaniem UNIX-a.

Na początku tego roku UI zdobyła punkt – ukazał się od dawna zapowiadany AT&T UNIX System V Release 4, czyli SVR4. Zamierzenie było ambitne, chodziło m.in. o połączenie w jeden spójny system wszystkiego, co sprawdziło się w kilku głównych współczesnych wariantach UNIX-a (SVR3.2, 4.3BSD z Berkeley, SunOS i XENIX). Ponadto USO opublikowała szczegółowy plan rozwoju tego systemu w najbliższych latach.

System opracowywany przez OSF ma być gotowy w przyszłym roku. Dziś wiadomo niewiele ponad to, że ma nazywać się OSF/1 i że jest wzorowany na UNIX – o podobnym eksperymentalnym systemie Mach z uniwersytetu Carnegie-Mellon oraz na AIX-ie. Dzięki koneksjom z Mach-em będzie to prawdopo-

dokończenie na s. 16

/etc

W Polskim Towarzystwie Informatycznym powstała Sekcja UNIX-owa. Inicjatywę utworzenia sekcji zgłosiła grupa pracowników Instytutu Podstaw Informatyki PAN. Członkowie grupy uczestniczyli w realizacji systemu IPIX (polskiej wersji systemu UNIX-owego). Zebranie inauguracyjne odbyło się 18 czerwca 1990. Ustalono tymczasowe zasady pracy Sekcji. Powstało kilka grup roboczych, m.in. grupa redakcyjna, grupa wymiany informacji o oprogramowaniu UNIX-owym oraz grupa słownika UNIX-owego. Adres organizatora: **Jan Walasek, Instytut Podstaw Informatyki PAN, skr. poczt. 22, 00–901 Warszawa (PKiN).**

• • •

Z jednego z pierwszych tegorocznych numerów miesięcznika UNIXWORLD dowiedzieliśmy się, że UNIX Software Operation, dział AT&T zajmujący się UNIX-em, jest do kupienia za 450 milionów dolarów (cena obejmuje zespół, znak firmowy i prawa autorskie do UNIX-a). Niestety, aktualny roczny budżet Sekcji UNIX-owej PTI wynosi około 50 dolarów. Nic – poza zrzutką – nie przychodzi nam do głowy.

supra

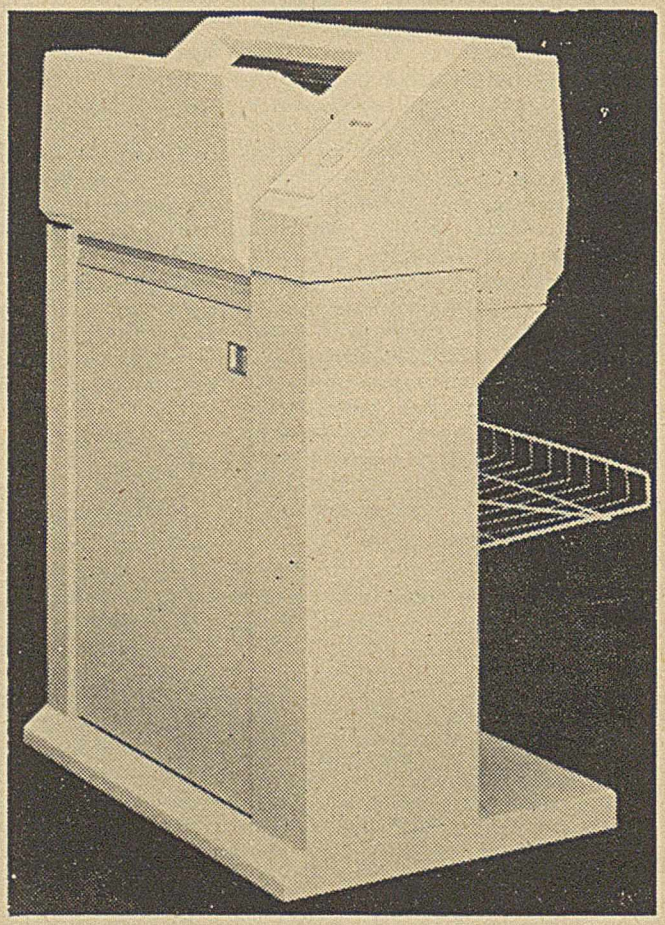
SUPRA SP. Z O.O.
ul. Trybunalska 54, 60-325 POZNAŃ
telefon: 674392, telefaks: 236498
teleks: 412966

OFERUJE ŚWIATOWY BESTSELLER

MANNESMANN
TALLY

DRUKARKĘ MOZAIKOWO-WIERSZOWĄ MT 645

- doskonała jakość druku
- niezawodność 7000 h MTBF
- duża szybkość 450 wierszy/min
- cicha praca
- bufor 27 KB
- liczba kopii 1+5
- pełna grafika 240 (dpi)
- polskie litery (opcja)
- wewnętrzny program testujący
- dwa interfejsy automatycznie przełączane
- podwójny system traktorów



MANNESMANN
TALLY

NAJPRAWDOPODOBNIEJ NAJLEPSZE
DRUKARKI NA ŚWIECIE

AUTORYZOWANY DYSTRYBUTOR DRUKAREK
MANNESMAN TALLY

supra

0/12/90

Ogłoszenie w INFORMATYCE bezbłędną szansą trafienia do właściwego odbiorcy!

Miesięcznik INFORMATYKA jest od 25 lat **jedynym** w Polsce czasopismem naukowo-technicznym poświęconym wszystkim aspektom technologii informatycznej (sprzęt komputerowy, oprogramowanie, zastosowania). Taka pozycja czasopisma sprawia, że INFORMATYKA dociera nie tylko do najszerszych kręgów środowiska profesjonalnego, ale również do wszystkich zainteresowanych aktualnymi możliwościami tej technologii.

Ogłoszenie w INFORMATYCE to najskuteczniejszy sposób pozyskania nowych odbiorców i rozszerzenia rynku zbytu. Gwarantujemy Państwu największe efekty przy najniższych nakładach finansowych, ponieważ **ceny ogłoszeń w INFORMATYCE są znacznie niższe** niż w innych czasopismach:

• cała strona	500 000 zł
• 3/4 strony	450 000 zł
• 2/3 strony	400 000 zł
• 1/2 strony	300 000 zł
• 1/3 strony	200 000 zł
• 1/4 strony	150 000 zł
• 1/8 strony	100 000 zł
• poniżej 1/8 strony	2 000 zł/cm^2

Dopłaty:

• za dodatkowy kolor	+ 30%
• za I i IV stronę okładki	+ 100%
• za II i III stronę okładki	+ 50%

Rabaty:

• za powtórzenie 3-6 razy	– 10%
• za powtórzenie ponad 6 razy	– 20%
• za teksty o charakterze informacji techniczno-handlowej (artykuły)	– 50%

Zapewniamy szybką i terminową realizację ogłoszeń!

Nasz adres: INFORMATYKA, 01–552 Warszawa, Pl. Inwalidów 10, p. 128, tel. 39-14-34

1990 PRINT „BASIC ma 26 lat"

dokończenie z III s. okładki

Dla wielu znajomość z językiem BASIC zakończyła się wiele lat temu. Pozostało w nich mniemanie o siermiężnej prostocie, nadmiernych ograniczeniach czy wręcz o zarażaniu początkujących programistów złymi nawykami. Istnienie łatwo dostępnego translatora Turbo Pascala czy modnego C łatwiej pozwalało rozstać się z BASIC-iem. Nastąpiło wręcz zwalczanie języka, aż do prawnych zakazów jego stosowania w nauczaniu początkowym. Dla szkół zaczęto propagować język LOGO, ale minęło już kilka lat i nie chce się on jakoś zadomowić. Z językiem programowania jest trochę podobnie jak z językiem ojczystym. Nowe słowa, zwroty są przyjmowane z dużymi oporami, natomiast powszechnie używane trudno jest zwalczyć. Szybciej zreformowały się środki masowego przekazu niż powszechnie zaakceptowano publikatory.

Język BASIC nie chciał jednak zniknąć z powierzchni informatycznej. Nadal ponad połowa nieprofesjonalnych programistów posługuje się nim, a nawet informatycy w chwili ostatecznej bitwy z komputerem sięgają po BASIC, jeżeli muszą szybko napisać prosty program testujący sprzęt lub fragment opornego algorytmu. Jednocześnie coraz lepsze podręczniki, biblioteki programów, a wreszcie kolejne, rozszerzone o instrukcje strukturalne, wersje tego języka pozwalają nowym użytkownikom komputerów na szybkie poznanie i zastosowanie poprawnych zasad programowania. I nie trzeba im wyjaśniać dlaczego 56 i 56.6 są liczbami innych typów. Znam też kilka dużych produktów programowych napisanych od początku do końca w języku BASIC i to niekoniecznie w jego najnowszej wersji.

Zresztą również języki Pascal i C stanęły pod pręgierzem braku elementów zorientowanych na programowanie obiektowe. Język BASIC tutaj też się spóźnia z wprowadzeniem pojęcia specyfikacji i implementacji modułu czy obiektu. Nie wątpię, że już niebawem pojawią się nowe jego wersje zawierające, podobnie jak C++, czy Turbo Pascal 5.5, elementy programowania obiektowego.

Dla języka BASIC pozostało jeszcze jedno ogromne pole zastosowań, które powinno zostać wreszcie zagospodarowane. Mając bowiem do dyspozycji nowoczesne strukturalne i modularne języki programowania musimy jeszcze ciągle pisać kostropate programy skryptów dla systemów operacyjnych (np. MS-DOS czy UNIX), czy procedur baz danych (np. dBASE) oraz arkuszy kalkulacyjnych. Trudno jest mi stwierdzić dlaczego nie sięgnięto tu przynajmniej po pierwotną wersję BASIC-a. Co więcej, nie widzę żadnych powodów, oprócz względów komercyjnych, braku ujednolicenia tych języków wbudowanych w – bądź co bądź – bardzo popularne i niezłe produkty.

Tak, jestem fanem języka BASIC. Widzę dużo jego zalet, ale również wad. Daleki jestem od nawoływania – „only you". Każdy z bardziej znanych, dotychczas istniejących języków ma swoje miejsce, nawet w dzisiejszej praktyce programowania. A wszystkich tych, którzy dawno lub nigdy nie mieli do czynienia z BASIC-iem zachęcam do próby poznania jego ostatnich wersji – może się spodobają!

Sądzę, że już w niedalekiej przyszłości w ślad za integracją świata nastąpi integracja proceduralnych języków programowania. Wszystkie one zaczną się tak do siebie upodabniać, że wreszcie ktoś, lub jakaś silna firma, zaproponuje nowy język programowania, zawierający ujednoliconą składnię i semantykę – jak najprostszą i najbardziej efektywną. Jaka będzie nazwa tego języka – nie ważne – ale na pewno nie będzie on zbyt trudny dla obecnych użytkowników języka BASIC lat dziewięćdziesiątych.

WACŁAW ISZKOWSKI

PS.1 Inspiracją do napisania tego tekstu był esej Billa Gatesa opublikowany w ubiegłorocznym listopadowym numerze miesięcznika BYTE.

PS.2 Zbitki poglądów i spostrzeżeń, tutaj prezentowanych, nie są oficjalnym stanowiskiem Redakcji ani żadnej innej organizacji. Ich prezentacja ma służyć pobudzeniu dyskusji na dany temat, nawet przez wzburzenie moich ewentualnych adwersarzy.

Wrzeszcz Z.: Język Lisp i elementy sztucznej inteligencji (1)

INFORMATYKA 1990, nr 9, s.1

Wprowadzenie w problematykę sztucznej inteligencji oraz ogólna charakterystyka struktury i funkcji języka programowania Lisp.

Wrzeszcz Z.: Lisp language and AI elements (1)

INFORMATYKA 1990, No. 9, p. 1

Introduction to AI problems and general characteristics of the Lisp programming language structure and functions.

Wrzeszcz Z.: Lisp-Sprache und Elemente der künstlichen Intelligenz (1)

INFORMATYKA 1990, Nr. 9, S. 1

Einführung zur Problematik der künstlichen Intelligenz und eine Charakteristik von Struktur und Funktionen der Lisp-Programmiersprache.

Górecki A.: Historyczna szansa polskiej informatyki

INFORMATYKA 1990, nr 9, s. 5

Sugestia przyspieszenia rozwoju informatyki w Polsce oraz możliwości zlikwidowania obecnego zapóźnienia w tej dziedzinie w stosunku do krajów przodujących.

Górecki A.: Historic chance of polish data processing

INFORMATYKA 1990, No. 9, p. 5

Suggestion for data processing acceleration in Poland and liquidation possibility of actual delay in comparison with leading countries in this area.

Górecki A.: Historische Chance der polnischen Informatik

INFORMATYKA 1990, Nr. 9, S. 5

Eine Suggestion der Beschleunigung von Informatikentwicklung in Polen und Beseitigungsmöglichkeit der jetzigen Verspätung im Vergleich mit führenden Ländern.

Zieliński K.: Rozproszone przetwarzanie w grafice i transputery

INFORMATYKA 1990, nr 9, s. 7

Rozpraszanie przetwarzania zadań graficznych oraz wykorzystanie tej metody w systemach graficznych budowanych przy użyciu transputerów.

Zieliński K.: Distributed processing in graphics and transputers

INFORMATYKA 1990, No. 9, p. 7

Processing distribution of graphic tasks and utilization of such method in graphic systems constructed with transputers.

Zieliński K.: Zerstreute Verarbeitung in Graphik und Transputeren

INFORMATYKA 1990, Nr. 9, S. 7

Zerstreuen von Verarbeitung der graphischen Aufgaben und Ausnutzung dieser Methode in graphischen Systemen, die mit Verwendung von Transputeren gebaut werden.

Cylkowski Z.: Pomiar czasu obliczeń na IBM PC

INFORMATYKA 1990, nr 9, s. 11

Charakterystyka sposobu mierzenia czasu obliczeń wykonywanych na komputerach IBM PC na podstawie programów napisanych w języku Turbo Pascal.

Cylkowski Z.: Computation time measurement on IBM PC

INFORMATYKA 1990, No. 9, p. 11

Characteristics of the method for time measurement of computations executed on IBM PC computers using programs written in Turbo Pascal language.

Cylkowski Z.: Messung der Rechenzeit auf IBM PC

INFORMATYKA 1990, Nr. 9, S. 11

Eine Charakteristik von Zeitmessungsmethode der auf IBM PC realisierten Berechnungen mit Anwendung von in der Turbo Pascal-Sprache geschriebenen Programmen.

Żychowski P.: Automatyczna standaryzacja słów tekstu w języku polskim

INFORMATYKA 1990, nr 9, s. 15

Charakterystyka budowy oraz wyniki stosowania algorytmu doprowadzającego słowa tekstu w języku polskim do ustalonej formy gramatycznej.

Żychowski P.: Automatized standarization of text words in polish language

INFORMATYKA 1990, No. 9, p. 15

Characteristics and application results of the algorithm which leads text words in polish language to certain grammatical form.

Żychowski P.: Automatisierte Standarisierung von Textworten in der polnischen Sprache

INFORMATYKA 1990, Nr. 9, S. 15

Eine Charakteristik und Ergebnisse von Anwendung des Algorithmus, der die polnische Textworte zu festgestellten grammatischen Form führt.

Muszyński M.: Fizyczne struktury systemów baz danych (2). Techniki mieszania

INFORMATYKA 1990, nr 9, s. 17

Druga część wprowadzania w dziedzinę fizycznych struktur systemów baz danych, zawierająca omówienie różnych technik mieszania danych.

Muszyński M.: Physical structures of database systems (2). Hashing techniques

INFORMATYKA 1990, No. 9, p. 17

Second part of an introduction to the field of physical database structures, which includes discussion of different hashing techniques.

Muszyński M.: Physische Strukturen von Datenbanksystemen (2). Die Mischungstechniken

INFORMATYKA 1990, Nr. 9, S. 17

Zweiter Teil einer Einführung im Bereich der physischen Strukturen von Datenbanksystemen, der eine Besprechung von verschiedenen Datenmischungstechniken umfasst.

Postół M.: Przemysłowe systemy czasu rzeczywistego (2). Struktura i funkcje biblioteki DIALOGUE

INFORMATYKA 1990, nr 9, s. 22

Druga część charakterystyki przemysłowych systemów czasu rzeczywistego, zawierająca omówienie struktury i funkcji oraz przykład zastosowania biblioteki DIALOGUE.

Postół M.: Industrial real-time systems (2). Structure and functions of the DIALOGUE library

INFORMATYKA 1990, No. 9, p. 22

Second part of characteristics of industrial real-time systems, which includes discussion of structure, functions and application example of the DIALOGUE library.

Postół M.: Industrielle Echtzeitsysteme (2). Struktur und Funktionen der DIALOGUE-Bibliothek

INFORMATYKA 1990, Nr. 9, S. 22

Zweiter Teil einer Charakteristik von industriellen Echtzeitsystemen, der eine Besprechung von Struktur und Funktionen der DIALOGUE-Bibliothek, sowie ein Beispiel ihrer Anwendung, umfasst.

1990 PRINT „BASIC ma 26 lat"

W 1962 roku profesor matematyki, Thomas Kurtz, zaproponował profesorowi John Kemeny, dziekanowi Dartmouth College, wprowadzenie czteroletniego kursu informatyki dla studentów tej uczelni. W trakcie projektowania nowego wielodostępnego systemu dla nowego komputera opracowali oni prosty język wysokiego poziomu – czerpiąc najlepsze wzorce z Fortranu i Algolu 60. W rezultacie 1 maja 1964 roku na dalekopisach przyłączonych do nowego komputera pojawił się znak zachęty READY>, wyrażający gotowość nowego języka do działania. Był to język BASIC (Beginner's All-purpose Symbolic Instruction Code), czyli kod symbolicznych instrukcji do wszystkich zastosowań dla początkujących.

W krótkim czasie język ten zaczął zyskiwać coraz większą popularność i został zaimplementowany na komputerach IBM, PDP-8, PDP-10, PDP-11, HP, CDC, ICL itp. W trakcie tych implementacji podstawowa wersja języka była rozszerzona o nowe konstrukcje. Brak standardu spowodował, że nowe elementy języka coraz bardziej różniły się między sobą pod względem składni i semantyki, co nie ułatwiało przenoszenia programów między różnymi instalacjami. Tak mijały lata sześćdziesiąte.

W 1972 roku dwóch studentów wydziału Elektroniki Politechniki Warszawskiej, mając jedynie przeznaczony dla studentów egzemplarz książki obu profesorów, rozpoczęło implementację interpretera języka na maszynie Odra 1204. Mieliśmy zielone pojęcie jak to należy zrobić, co gorsza nie tylko my, chociaż na tej maszynie działał chyba jeden z najlepszych kompilatorów Algolu-60, napisany przez J. Szczepkowicza z Wrocławia. Była to jednak praca dyplomowa, a więc musiała się zakończyć sukcesem. Interpreter BASIC-a działał. Gorzej było z komputerem, który z czasem został wycofany z eksploatacji.

Później zaczęliśmy eksploatować BASIC na komputerze CYBER-72. Mając do dyspozycji tylko jeden terminal, uczyliśmy studentów korzystania z konwersacyjnego interpretera tego języka. Fakt, że teksty większych programów wprowadzaliśmy do pliku z kart. Porównując jednak dwie grupy, z których jedna uruchamiała programy w Fortranie w trybie wsadowym, a druga te same programy w BASIC-u w trybie konwersacyjnym, szybko stwierdziliśmy znacznie większą efektywność tej drugiej metody. Niestety praca dużej grupy studentów z jednym terminalem wymagała ogromnej ekwilibrystyki. Jedynym efektem tych działań pozostał centralny skrypt opisujący po polsku język BASIC.

Nastał rok 1974 – początek ery mikrokomputerowej. Język BASIC, już nieco zapomniany w uproszczonej postaci był stosowany w kalkulatorach. Naturalnym było więc, że w 1975 r. język ten rozpoczął drugą młodość, wraz z pojawieniem się pierwszych mikroprocesorów mających do dyspozycji zaledwie 4KB pamięci operacyjnej. Paul Allen i Bill Gates napisali pierwszy translator interpreter BASIC-u dla mikrokomputera MITS Altair 8800, zastępując READY> krótszym OK> (dla oszczędności trzech bajtów pamięci operacyjnej). Później kariera BASIC-a potoczyła się już wartko. Powstał Tiny BASIC dla Wanga, Radio Shack Level 1 BASIC, Apple Integer BASIC Steve'a Woźniaka oraz Gordona Eubanksa BASICe i CBASIC. Ukoronowaniem tych wysiłków i standardem przemysłowym stał się Microsoft BASIC, wywodzący swą nazwę od firmy założonej przez Billa Gatesa.

Wzorcowy język BASIC był bardzo prosty, takie też były jego pierwsze implementacje dla mikrokomputerów. Pozwalał on na bardzo szybkie poznanie kilku prostych instrukcji i prawie natychmiastowe napisanie prostych programów. Jednak już w niedługim czasie zaczął stwarzać problemy przy próbach zapisu nieco bardziej złożonych algorytmów. Powodowały je: ograniczona do dwóch znaków długość identyfikatorów, brak jawnego deklarowania zmiennych, numerowanie linii, prosta instrukcja cyklu i instrukcja warunkowa oraz brak podprogramów z własnymi lokalnymi zmiennymi.

W 1974 roku pod auspicjami ANSI rozpoczął się pod przewodnictwem Thomasa Kurtza proces opracowywania i zatwierdzania normy języka. Proces ten się jeszcze nie zakończył, chociaż powstały już kolejne wersje raportów języka. W aktualnej wersji elementy grafiki oparto na standardzie GKS; dodano też proste elementy współbieżności procesów, umożliwiające stosowanie języka do oprogramowywania systemów sterowania w czasie rzeczywistym.

W chwili obecnej istnieje wiele różnych translatorów języka. Kilku znanych producentów oprogramowania, właściwie już na wyścigi, wprowadza na rynek coraz to nowsze wersje zintegrowanych pakietów z językiem BASIC w roli głównej. I tak Microsoft oferuje już wersje 4.5 swojego QuickBASIC-a. Konkurujący z nim Borland po wprowadzeniu wersji 1.1 TurboBASIC-a zwrócił prawa jego autorowi, który w firmie Spectra Publishing Software właśnie anonsuje na rynku PowerBASIC w wersji 2.0. Oprócz tego dostępne są nowe wersje TrueBASIC, z firmy Kurtz, oraz CBASIC-86, z Digital Research, a także BetterBASIC – już nie pamiętam skąd. Jednocześnie kilka innych firm oferuje dla języka BASIC biblioteki procedur, ułatwiające konstruowanie złożonych programów. Przykładem takiej biblioteki są pakiety ProBAS, ProREF, Prô... oferowane przez Hammerly Computer Service.

I wreszcie koronnym dowodem na popularność języka BASIC jest umieszczenie w pamięci RAM interpretera języka GW BASIC w oryginalnych mikrokomputerach IBM PS/2 (sic!).

dokończenie na s. 31

PRZEDSIĘBIORSTWO
ZASTOSOWAŃ INFORMATYKI

meditronik

oferuje:

- systemy mikrokomputerowe
- programy aplikacyjne dla różnych dziedzin gospodarki (na życzenie wysyłamy katalog)
- poszukiwane komponenty elektroniczne
- interfejs do kamery video (opc. CCD) z bogatą biblioteką oprogramowania
- emulator Z80
- tester układów scalonych i pamięci
- programator BPROM
- asynchroniczny procesor komunikacyjny
- konwerter RS-232 – Centronics

instaluje:

- połączenia międzykomputerowe (XT/AT – ODRA/RIAD/IBM)
- systemy sieciowe (NOVELL)
- systemy wielodostępne (SCO Xenix 286, 386, Unix System V)

Jeżeli jesteś autorem oryginalnego programu aplikacyjnego
- skontaktuj się z nami, będziemy pośredniczyć w sprzedaży Twojego programu, dbając o ochronę Twoich praw autorskich!

Nasz adres:

00–194 Warszawa, ul. Dzika 4
tel. (02) 635–22–63, 635–22–64,
fax (02) 635–21–95
tlx 816075 medi pl

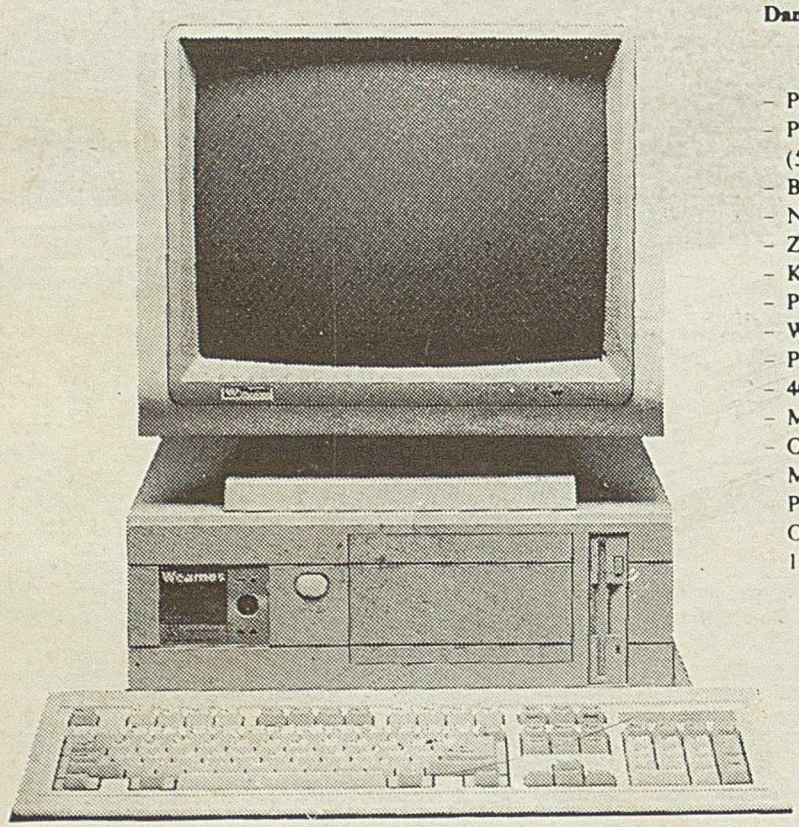
EO/605/89

KOMPUTER
Z WBUDOWANĄ PRZYSZŁOŚCIĄ

Komputer zaprojektowany przez czołowego światowego producenta mikrokomputerów – firmę ADVANCED LOGIC RESEARCH i wyprodukowany przez firmę WEARNES TECHNOLOGY.

Komputer, którego możliwości i cena oszałamiają konkurencję.

Komputer, który nie zestarzeje się, dzięki możliwości rozbudowy do architektury 386SX i 486.



Dane techniczne:

SERIA WEARNES BOLDLINE „M"

- Procesor 80286 – 12.5 Mhz
- Pamięć 1 MB RAM, możliwość rozbudowy do 16 MB (5 MB na płycie głównej)
- BIOS Phoenix
- Napęd dyskietek 3,5" 1,44 MB
- Zasilacz 110 Watt
- Klawiatura 101 klawiszy
- Podstawka dla koprocesora matematycznego 80287
- Wbudowany sterownik dyskowy z przeplotem 1:1
- Port szeregowy i równoległy
- 40 MB dysk sztywny
- Mozliwość korzystania z EMS 4.0
- Obudowa typu „compact"
- Miejsce na dwa napędy 5,25" o wysokości 1/2
- Podręcznik i dyskietka z programem konfiguracyjnym
- Opcjonalna rozbudowa do 386SX i 486
- 12-miesięczna gwarancja

BOLDLINE COMPUTERS
GRUPA MICOMP-TECH
Biuro Informacji Techniczno-Handlowej
ul. Astrów 7, 40–045 Katowice
telefon i telefaks: 518-628
teleks: 315687 COMP PL

DYSTRYBUTORZY:
PTH „TECHMEX"
43–300 Bielsko-Biała
ul. M. Curie Skłodowskiej 13
tel.: 42-198, 47-555, telefaks: 47-624,
teleks: 35325

EO/1123/89